UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS

Francisco Ambrosio Garcia

Segmentação do caminhar humano via modelos de Markov

São Carlos

2020

Francisco Ambrosio Garcia

Segmentação do caminhar humano via modelos de Markov

Monografia apresentada ao Curso de Engenharia Elétrica com Ênfase em Eletrônica, da Escola de Engenharia de São Carlos da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Engenheiro Eletricista.

Orientador: Prof. Dr. Marco Henrique Terra

São Carlos

2020

Ficha catalográfica elaborada pela Biblioteca Prof. Dr. Sérgio Rodrigues Fontes da EESC/USP com os dados inseridos pelo(a) autor(a).

Garcia, Francisco Ambrosio
G216s Segmentação do caminhar humano via modelos de Markov / Francisco Ambrosio Garcia; orientador Marco Henrique Terra. São Carlos, 2020.

Monografia (Graduação em Engenharia Elétrica com ênfase em Eletrônica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2020.

1. Segmentação do Caminhar Humano. 2. Processos Estocásticos. 3. Modelos de Markov. 4. Unidades de Medidas Inerciais. I. Título.

Eduardo Graziosi Silva - CRB - 8/8907

FOLHA DE APROVAÇÃO

Nome: Francisco Ambrosio Garcia

Título: "Segmentação do caminhar humano via modelos de Markov"

Trabalho de Conclusão de Curso defendido e aprovado em__04__/_12___/_2020__,

com NOTA__Dez____(10 , 0), pela Comissão Julgadora:

Prof. Titular Marco Henrique Terra - Orientador SEL/EESC/USP

Prof. Associado Adriano Almeida Gonçalves Siqueira - SEM/EESC/ USP

Dr. Juan Carlos Perez Ibarra - Programa de Pós-doutoramento EESC/USP

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Associado Rogério Andrade Flauzino

Este trabalho é dedicado à minha família, que sempre me apoiou e envidou todos os esforços para que eu me desenvolvesse como ser humano. Muito obrigado.

AGRADECIMENTOS

Em primeiro lugar agradeço aos meus pais Maria Inês e José Eduardo por terem fornecido a base sólida para minha existência. Agradeço igualmente meu irmão Antonio que estimo imensamente. À minha prima Luana que me apoiou nos momentos mais difíceis e é fonte de admiração. Agradecimento especial à minha tia Ana e ao meu tio Mario que me forneceram apoio imensurável.

Muito obrigado aos meus amigos da Elétrica: Thamer, Igor, Rafael Fenerick, Rafael Arone, Rafael Bagagli, Daniel, Paulo, Remo, Raul, Rogério, Leonardo, Vinícius, Jorge, Ricardo, Larissa, Alisson que compartilharam os momentos da graduação.

Um agradecimento particular aos meus amigos Gabriel, Paula, Felipe, Leonardo, Vinícius, Marcos Paulo, Guilherme, Caio, Yan, Marcos Saito, Liao Zihao, Fabrice, que estiveram sempre presentes nas maravilhas e dificuldades dos estudos no exterior.

Muito obrigado aos meus amigos de longa data Leonardo, Pedro, Felipe, Nestor, Igor, Hugo, por estarem ao meu lado durante todos esses anos.

Agradeço ao meu orientador Prof. Dr. Marco Henrique Terra por ter me apresentado o universo da pesquisa em sua área de domínio, pelas oportunidades de trabalho, orientação e inclusão na equipe. Ao Juan Carlos Perez Ibarra por compartilhar seu vasto conhecimento e pelos comentários valiosos. Também agradeço ao Christoph M. Mitschka que foi um grande incentivador durante meus primeiros anos de graduação. Aos colegas do laboratório LASI, sempre dispostos a ajudar. À Escola de Engenharia de São Carlos da Universidade de São Paulo e à CentraleSupélec, pela estrutura, oportunidades, laboratórios e ensino de qualidade que contribuíram para minha formação.

"Um homem precisa viajar para lugares que não conhece para quebrar essa arrogância que nos faz ver o mundo como o imaginamos, e não simplesmente como é ou pode ser; que nos faz professores e doutores do que não vimos, quando deveríamos ser alunos, e simplesmente ir ver."

Amyr Klynk

RESUMO

GARCIA, F. A. **Segmentação do caminhar humano via modelos de Markov**. 2020. 83p. Monografia (Trabalho de Conclusão de Curso) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2020.

A segmentação do caminhar humano pode auxiliar no acompanhamento de pacientes em tratamentos de reabilitação motora e é crucial para o controle de sistemas robóticos de suporte à marcha ou de reabilitação. Métodos já existentes realizam a detecção acurada de fases do caminhar em indivíduos com padrão de marcha normal. Recentemente avanços foram obtidos também em padrões de marcha alterados por alguma debilidade. Entretanto grande parte das abordagens demanda equipamentos disponíveis apenas em laboratórios especializados. Neste trabalho é realizada a segmentação da marcha humana via modelos de Markov utilizando sensores inerciais. Na segunda parte do trabalho são empregados *smartphones* para o treinamento, em substituição ao equipamento especializado. Dois modelos foram implementados, o primeiro baseado em uma cadeia de Markov e o segundo em um modelo oculto de Markov. Para o segundo as fases foram escolhidas de forma a possuírem sentido físico interpretável e foram utilizados sensores inerciais existentes em um *smartphone*. O treinamento foi realizado com dados de apenas um indivíduo por meio de uma abordagem baseada em regras. Em seguida os estados do caminhar humano foram decodificados pelo algoritmo de Viterbi, tanto em pós-processamento quanto em tempo real. O método em tempo real consistiu em limitar o processamento a uma janela deslizante de tamanho fixo. Por fim demonstrou-se uma aplicação que consiste em mensurar a duração dos passos e de fases. Obteve-se uma performance satisfatória em pós-processamento (F_1-*score* de 0.96), comparável aos métodos reportados na literatura. Entretanto a performance em tempo real (F_1-*score* de 0.55) foi inferior a outros trabalhos. Em especial constatou-se uma taxa de inserções elevada e um atraso inerente ao janelamento. Verificou-se que é possível realizar o treinamento com o emprego de um *smartphone*. Os resultados sugerem que uma quantidade bastante limitada de dados pode ser suficiente para treinar um modelo satisfatório específico para um indivíduo, porém tal hipótese exige melhor investigação em trabalhos futuros.

Palavras-chave: Segmentação do Caminhar Humano. Processos Estocásticos. Modelos de Markov. Unidades de Medidas Inerciais.

ABSTRACT

GARCIA, F. A. **Markov models-based gait segmentation**. 2020. 83p. Monografia (Trabalho de Conclusão de Curso) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2020.

Human gait segmentation may help monitor patients undergoing rehabilitation treatments and is crucial for the control of robotic systems designed to support walking or rehabilitation. Several approaches have been proposed, some of which accurately detect human gait phases in real-time for healthy subjects. Recently a great performance was also obtained for impaired gait. However, a large proportion of the existing methods require devices that are only available in specialized laboratories. This work describes human gait segmentation via Markov models using inertial sensors. In the second part smartphones were used to train the model, in substitution to the specialized equipment. Two models were implemented. The first one is based on a Markov chain, whereas the second is a hidden Markov model. For the second one the phases were chosen to have physical meaning and the inertial sensors from a smartphone were used to train the model. The training was done following a rule-based approach with data collected from a single healthy subject. Afterwards the gait phases were decoded using the Viterbi algorithm, both in post-processing and in real-time. The real-time approach consisted in limiting the computation to a sliding window with fixed length. Finally, an application of measuring the duration of the steps and gait phases is demonstrated. The post-processing algorithm showed a performance (F_1-*score* of 0.96) comparable to other methods available in the literature. However, the performance in real-time (F_1-*score* of 0.55) was inferior to other methods. In particular, it produced a high insertion rate and a delay related to the windowing. The smartphone was suitable for the training. The results suggest that a very limited dataset may be enough to train a satisfactory model for a single subject, although this hypothesis requires further investigation in future work.

Keywords: Gait Segmentation. Stochastic Processes. Markov Models. Inertial Measurement Units.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

MPT	Matriz de Probabilidades de Transição (ou matriz de transição)
IMU	Unidade de Medidas Inerciais

LISTA DE SÍMBOLOS

$\mathbb{N}_0$	Números inteiros não negativos
$\mathbb{R}$	Números reais
$\mathbb{C}$	Números complexos
$\mathbb{C}^{\mathbb{N}}$	Espaço vetorial complexo n-dimensional
$(\Omega, \mathcal{F}, \mathcal{P})$	Espaço de probabilidade
X	Processo Estocástico
$\mathbf{P}$	Matriz de Probabilidades de Transição de uma Cadeia de Markov
$\mathbf{A}$	Matriz de Probabilidades de Transição de um Modelo Oculto de Markov
$\mathbb{P}\{A\}$	Probabilidade do evento A
$\boldsymbol{\pi}$	Vetor estado inicial de um Modelo Oculto de Markov
$\boldsymbol{\phi}$	Matriz de emissão de um Modelo Oculto de Markov
S	Espaço de estados
p_{ij}	Probabilidade de transição do estado i ao estado j em uma Cadeia de Markov
a_{ij}	Probabilidade de transição do estado i ao estado j em um Modelo Oculto de Markov

SUMÁRIO

1 INTRODUÇÃO

A detecção de fases da marcha humana é crucial para algumas aplicações, por exemplo tratamentos de reabilitação motora como estimulação elétrica funcional, controle de sistemas robóticos de suporte à marcha ou reabilitação e acompanhamento de pacientes em tratamentos de recuperação motora. Muitas vezes, o sistema de detecção deve operar em tempo real com baixo atraso e elevada acurácia. Assim, o problema da segmentação do caminhar humano, possivelmente em tempo real, tem ampla relevância.

Diversas soluções foram propostas. Em geral elas podem ser divididas em dois grupos: soluções baseadas em regras (análise no domínio temporal, *thresholds*, valores de pico, cruzamento de zero, derivadas, médias, entre outras métricas), e soluções baseadas em aprendizado de máquina (Modelos Ocultos de Markov, redes neurais artificiais, aprendizagem profunda, entre outras) (VU et al., 2020).

As técnicas baseadas em regras possuem a vantagem de serem de simples implementação e não demandarem um treinamento prévio do modelo, como acontece com modelos de aprendizagem de máquina. Além disso, o poder de processamento exigido para verificar as regras é baixo, o que possibilita a aplicação em tempo real. Entretanto, a escolha das regras exige um conhecimento sobre os padrões de caminhada e das características dos sinais medidos.

Por outro lado, métodos baseados em aprendizagem de máquina possuem a vantagem de não exigirem um conhecimento especialista nos padrões dos sinais capturados. Dentro desse contexto, na sequência de artigos de Mannini *et al.*, os autores relatam que o algoritmo desenvolvido utilizando Modelos Ocultos de Markov apresentou precisão maior que Modelos de Misturas Gaussianas (*Gaussian Mixtures*), Máquinas de Vetores de Suporte (*Support Vector Machines*) e Análise Discriminante Linear (*Linear Discriminant Analysis*), e atraso menor que métodos baseados em regras.

A segmentação da marcha pode ocorrer de duas formas: em pós-processamento ou em tempo real. A detecção em tempo real é crucial para por exemplo o controle de exoesqueletos, porém apresenta alguns desafios. Os algoritmos devem ser causais e não podem ser computacionalmente custosos. Observa-se que grande parte dos métodos existentes na literatura não são viáveis em tempo real. Além disso, os desafios são ainda maiores em indivíduos com alguma debilidade na marcha visto que a acurácia da maioria dos métodos decai significativamente (Pérez-Ibarra et al., 2018).

Os sinais captados para se estimar os estados e eventos do caminhar podem provir de diversos sensores, como sensores de pressão e botões na sola do pé, plataformas com sensores de força, sistemas optoeletrônicos, acelerômetros, giroscópios, Unidades de Medidas

Inerciais, sensores de atividade elétrica nos músculos, entre outros (VU et al., 2020).

Os sensores de pressão, força e sistemas optoeletrônicos são considerados o padrão ouro para a detecção dos eventos da marcha humana, porém não são apropriados para aplicações autônomas e que devam continuar funcionando por um longo prazo. Os sensores inerciais têm demonstrado bons resultados para suprir essa dificuldade, pois têm uma vida útil longa e custo reduzido. Além disso, hoje eles estão presentes na grande parte dos aparelhos celulares modernos (*smartphones*), sendo utilizados para por exemplo orientar a tela corretamente quando o celular é rotacionado. Em geral, um sensor inercial é composto por alguns acelerômetros e giroscópios dispostos em diferentes direções, que determinam os graus de liberdade suportados.

O presente trabalho tem objetivo de detectar fases (ou estados) do caminhar humano, tanto em pós-processamento quanto em tempo real com o emprego de sensores inerciais. Optou-se por utilizar modelos de Markov, sendo um deles um Modelo Oculto de Markov.

Para isso, primeiramente o caminhar foi analisado em diferentes cenários reais, como em rampas ou escadas, o que foi de grande valia para o projeto $2012/14074-3$ da FAPESP. Nele, foi desenvolvido um controlador Markoviano para o exoesqueleto Exo-Kanguera, utilizado em tratamentos de reabilitação motora. Um elemento crucial de tal controlador é a Matriz de Probabilidades de Transição (MPT), que, no caso acima, refere-se ao caminhar humano.

Essa matriz fundamentalmente descreve as probabilidades dos possíveis estados futuros do caminhar com base no estado atual, graças ao conhecimento de dados prévios. Esses dados foram coletados neste trabalho e uma MPT do caminhar humano foi derivada com base no modelo do exoesqueleto. No projeto citado, o modelo Markoviano da marcha humana é baseado em divisões em setores das articulações. Os estados desse modelo não possuem sentido físico *a priori*, sendo apenas uma subdivisão matemática e não são intuitivamente interpretáveis.

Em seguida, um segundo modelo foi derivado, baseado em Modelos Ocultos de Markov. Os estados foram escolhidos de forma a terem sentido físico interpretável. Após a fase de treinamento, o modelo foi utilizado para decodificar fases da marcha tanto em pós-processamento quanto em tempo real. Em pós-processamento foi utilizado o algoritmo de Viterbi convencional, e em tempo real aplicou-se o algoritmo de Viterbi limitado a uma janela deslizante de tamanho fixo. O desempenho em pós-processamento foi satisfatório e comparável a outros métodos relatados na literatura. Entretanto, o desempenho em tempo real foi inferior a outros trabalhos já existentes. Em particular, constatou-se uma alta taxa de inserção de ciclos completos da marcha dentro de um ciclo real. As principais publicações que servem de base para a derivação deste segundo modelo são (MANNINI; SABATINI, 2011) e (MANNINI; GENOVESE; SABATINI, 2013).

As principais contribuições deste trabalho são a própria implementação dos métodos de segmentação da marcha bem como as matrizes obtidas e a utilização de um *smartphone* para a captura dos dados. Além disso, os resultados obtidos em pós-processamento sugerem a hipótese de que um pequeno conjunto de dados pode ser suficiente para treinar um modelo da marcha satisfatório específico para um único indivíduo. Contudo, é necessária uma investigação maior para elucidar essa hipótese.

2 MÉTODOS

2.1 Métodos de segmentação do caminhar humano

O caminhar humano pode ser segmentado em diferentes níveis de granularidade, variando de dois estados até oito estados. Os métodos considerados em (VU et al., 2020) apresentaram acurácia de 100% quando a marcha foi segmentada em apenas duas fases. Conforme o número de subdivisões aumenta, a acurácia dos métodos diminui.

Diversos algoritmos existem para identificar eventos e fases da marcha humana. Pode-se dividi-los em dois grupos. O primeiro grupo corresponde aos algoritmos baseados em regras que tentam identificar características específicas das formas de onda de grandezas medidas durante o caminhar, bem como valores dessas grandezas durante as transições de fases.

Por exemplo, (CATALFAMO; GHOUSSAYNI; EWINS, 2010) propuseram um método baseado regras envolvendo mínimos e cruzamentos por zero do sinal de um giroscópio posicionado na canela para detectar dois eventos: contato inicial do pé com o chão e o momento em que o pé perde o contado com o chão. Foram considerados sete indivíduos com padrão de marcha normal em terrenos diversos. Nesse caso a taxa de detecção foi superior a 98%. Vale ressaltar que esse algoritmo é viável para ser aplicado em tempo real. A média do valor absoluto do atraso ou adiantamento para o segundo evento foi menor que 75 ms.

Já no trabalho de (MARIANI et al., 2013) diversas regras foram avaliadas com o objetivo de segmentar a marcha em quatro fases. Utilizaram-se sensores inerciais posicionados sobre o pé, e os algoritmos foram testados em indivíduos com ou sem debilidades na marcha. As melhores regras obtidas para cada um dos quatro eventos que indicam as transições entre as fases tiveram a média do erro absoluto do atraso menor que 42 ms.

(LEE; PARK, 2011) desenvolveram um algoritmo para detectar três eventos: *Initial Contact* (IC), *Mid-Swing* (MS) e *End-Contact* (EC) utilizando a velocidade angular na direção médio-lateral de um giroscópio posicionado na canela. Os eventos IC e EC são equivalentes aos eventos *Heel-Strike* (HS) e *Toe-Off* (TO) apresentados na seção 2.4.2. Esse método é baseado na sequência de dois picos negativos seguidos de um pico negativo do sinal. Ele também é aplicável ao sinal da velocidade angular na direção médio-lateral de um giroscópio posicionado sobre o pé, pois a curva também apresenta uma sequência de dois picos negativos seguidos de um pico positivo. O algoritmo de Lee *et al.* foi utilizado aqui como padrão de comparação para os algoritmos implementados.

Existem também soluções baseadas em aprendizado de máquina (Modelos Ocultos de Markov, redes neurais artificiais, aprendizagem profunda, algoritmos genéticos, entre

outros). A seguir são apresentados alguns trabalhos relevantes nessa categoria em que a marcha é segmentada em quatro fases.

(Pérez-Ibarra; Siqueira; Krebs, 2020) propuseram um método que utiliza classificadores lineares para detectar as transições entre quatro fases da marcha em tempo real, sendo uma generalização dos algoritmos baseados em regras. A performance foi comparável a outros métodos, com a vantagem de não se requerer conhecimento especializado prévio sobre as curvas e valores específicos nas transições dos estados.

(MANNINI; SABATINI, 2011) utilizaram um Modelo Oculto de Markov. A especificidade e sensibilidade relatada é superior a 95%, e a média do erro absoluto do atraso de aproximadamente 35 ms, sendo que a segmentação se deu em pós-processamento. Em um trabalho posterior os autores adaptaram o método para ocorrer em tempo real (MANNINI; GENOVESE; SABATINI, 2013).

O presente trabalho reproduz grande parte das ideias e métodos das duas publicações de Mannini *et al.* citadas no parágrafo anterior. Entretanto, o método para a segmentação da marcha em tempo real difere um pouco, visto que aqui foi utilizado o algoritmo de Viterbi convencional em uma janela deslizante de tamanho fixo. Além disso, tanto em pós-processamento quanto em tempo real aqui não foi considerada uma regra adicional que proíbe ciclos de marcha muito curtos. Dessa forma, a base teórica é dada a seguir.

2.2 Processos de Markov

Os Processos de Markov possuem uma estrutura matemática que pode ser aplicada a diversos problemas. Visto que a marcha humana será modelizada como uma Cadeia de Markov, caso particular dos Processos de Markov, faz-se necessário formulá-la matematicamente. Deve-se ainda compreender como uma Cadeia de Markov pode ser implementada computacionalmente e quais são os principais algoritmos de treinamento da Cadeia.

Dessa forma, proceder-se-á pela definição de um Processo Estocástico:

Definição 1 (Processo Estocástico)**.** *Seja t uma variável escalar real, que assume valores em T, ou seja, $t \in T \subset \mathbb{R}$. Seja S um conjunto denominado* ***espaço de estados****. Nas aplicações reais, temos com frequência $S = \mathbb{R}, \mathbb{C}, \mathbb{C}^{\mathbb{N}}$ ou um conjunto enumerável. Um processo estocástico (ou aleatório) é uma família de variáveis aleatórias sobre um mesmo espaço de probabilidade $(\Omega, \mathcal{F}, \mathcal{P})$ indexada por T e cujas variáveis aleatórias assumem valores em S. Cada uma dessas variáveis aleatórias é representada por X_t. O processo estocástico é denotado por $X = (X_t, t \in \mathbb{T})$.*

Neste projeto, t será a variável de tempo que mais adiante será discretizado para permitir implementação computacional do modelo. Fixando-se $\omega \in \Omega$, define-se a realização

ou **trajetória de um processo estocástico** como a aplicação que associa cada instante de tempo t ao valor $X_t(\omega)$.

Um processo estocástico é dito **Processo de Markov** se e somente se a distribuição de probabilidade condicional dos estados futuros dado os estados passados e presente depende apenas do estado presente. Dessa forma, a melhor estimação que se pode fazer dos estados futuros do processo pode ser calculada tendo como base apenas o estado presente. Precisamente, temos a seguinte definição abaixo, que pode ser encontrada em (NIELSEN, 2009):

Definição 2 (Processo de Markov)**.** *$X = (X_t, t \in \mathbb{T})$ é um processo de Markov se para todo instante $t \in T$ e $s \in T$, $s > t$, a lei de probabilidade condicional de X_s sabendo as variáveis X_u, $u \leq t$, depende unicamente de X_t. Ou seja, para qualquer $C \in S$, a lei $\mathbb{P}\{X_s \in C | X_u, u \leq t\}$ é uma função exclusivamente de t, s, X_t e C.*

As Definições 1 e 2 descrevem os casos gerais em que o tempo é contínuo e o espaço de estados S pode ser um conjunto não enumerável. Discretizando-se o tempo de forma que $t \in \mathbb{N}_0 = \{0, 1, 2, ...\}$, e assumindo que o espaço de estados é um conjunto enumerável cujos elementos são denotados por $i_0, i_1, ..., i_{n-1}, i_n, j$, definimos uma **Cadeia de Markov**. Essa definição pode ser encontrada em (PARDOUX, 2006) ou em livros básicos sobre Processos Estocásticos.

Definição 3 (Cadeia de Markov)**.** *Seja $X = (X_n, n \in \mathbb{N}_0)$ um processo em tempo discreto cujo espaço de estados é S. X é uma cadeia de Markov se para todo $n \in \mathbb{N}_0$ e para todos $i_0, i_1, ..., i_{n-1}, i_n, j \in S$, tem-se:*

$$\mathbb{P}\{X_{n+1} = j | X_n = i_n, X_{n-1} = i_{n-1}, ..., X_0 = i_0\} = \mathbb{P}\{X_{n+1} = j | X_n = i_n\} \tag{2.1}$$

Além disso, X é dita **Cadeia de Markov Homogênea** se o membro da direita da Equação 2.1 não depende de n.

Neste ponto pode-se definir a **probabilidade de transição** do estado i no instante n ao estado j como $p_{ij} = \mathbb{P}\{X_{n+1} = j | X_n = i\}$. Vale ressaltar que para uma cadeia de Markov homogênea essa probabilidade independe de n.

A partir das probabilidades de transição, constrói-se a **Matriz de Probabilidades de Transição** (MPT) (ou matriz de transição) $\mathbf{P} = [p_{ij}]_{(i,j) \in S^2}$, onde cada p_{ij} é colocado na linha i e coluna j. A matriz $\mathbf{P}$ é estocástica, ou seja, a soma dos termos de qualquer uma de suas linhas é igual a 1.

$$\forall i \in S, \quad \sum_{j \in S} p_{ij} = 1. \tag{2.2}$$

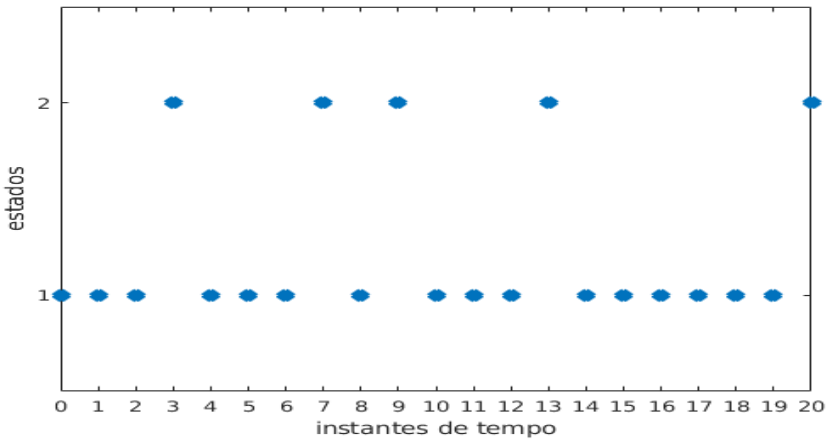
2.2.1 Exemplo simples de um Processo de Markov Homogêneo

Considere um sistema com dois estados $S = \{\theta_1, \theta_2\}$, que inicia no estado θ_1 em $t = 0$ e realiza saltos durante $t = (1, \ldots, 20)$. Em cada instante, se o sistema estiver no estado θ_1, ele tem 70% de chance de permanecer no mesmo estado e 30% de chance de saltar para o estado θ_2. Se o sistema estiver no estado θ_2, ele necessariamente retorna para o estado θ_1 no instante de tempo seguinte. Considera-se que ao final o sistema retorna ao estado θ_1. Com base nessas probabilidades pode-se construir a matriz de transição $\mathbf{P}_{teórica}$:

$$\mathbf{P}_{teórica} = \begin{bmatrix} 0.7 & 0.3 \\ 1 & 0 \end{bmatrix} \tag{2.3}$$

A seguir, realiza-se uma simulação em ambiente MATLAB do sistema, e mapeia-se a realização da cadeia de Markov, como mostrada na Figura 1. O código utilizado para a simulação encontra-se no Apêndice.

Figura 1: Sistema simples com dois estados Markovianos



Fonte: O Autor (2019)

Agora podemos verificar, através de uma contagem dos resultados da simulação, a frequência com que o sistema salta do estado θ_1 para o mesmo estado θ_1 e definir esse evento como p_{11}, e de θ_2 para θ_2 como p_{22}.

Além disso, denota-se a frequência com que o sistema salta de θ_1 para θ_2 como p_{12}, e o contrário como p_{21}. Finalmente, dividimos os saltos pelo número total de ocorrência de cada estado individual. Neste exemplo, o estado θ_1 aparece 16 vezes e o estado θ_2 aparece 5 vezes. Com esses dados calculam-se as probabilidades de transição:

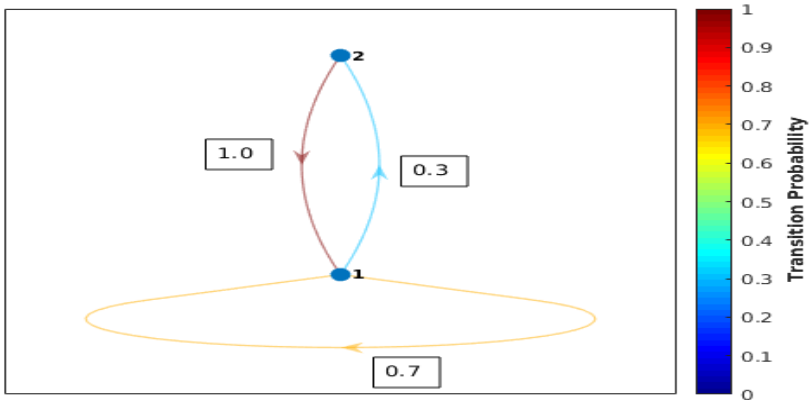
$$p_{11} = \frac{11}{16} = 0.6875 \qquad p_{12} = \frac{5}{16} = 0.3125,$$
$$p_{21} = \frac{5}{5} = 1 \qquad p_{22} = \frac{0}{5} = 0,$$

Dessa forma obtemos de forma empírica a MPT $\mathbf{P}_{empírica}$:

$$\mathbf{P}_{empírica} = \begin{bmatrix} 0.6875 & 0.3125 \\ 1 & 0 \end{bmatrix} \tag{2.4}$$

Pode-se representar a cadeia de Markov por um grafo direcionado em que os nós correspondem aos estados e as arestas às transições entre eles, cujos pesos são as probabilidades de transição. Esse grafo recebe o nome de **diagrama de estados**. Na Figura 2 é mostrado o diagrama de estados da cadeia referente à Equação 2.3.

Figura 2: Diagrama de estados do sistema simples



Fonte: O Autor (2019)

2.3 Modelo Oculto de Markov

Como discorrem (MANNINI; SABATINI, 2011), os Modelos Ocultos de Markov são amplamente utilizados em visão computacional para o reconhecimento de gestos. Além disso, tais modelos oferecem a possibilidade de segmentar a marcha humana utilizando dados coletados em velocidades e inclinações do terreno variadas. Nesse trabalho, os pesquisadores dividiram o caminhar em quatro estados e treinaram uma rede a partir de sinais provindos de um giroscópio monoaxial posicionado sobre peito do pé e orientado na direção médio-lateral, ou seja, apontando do centro para a lateral do corpo.

Seguindo esse raciocínio, observa-se que utilizar Modelos Ocultos de Markov para segmentar a marcha humana é apropriado aos fins deste projeto.

Dessa forma, definem-se os modelos ocultos de Markov da seguinte maneira: (BISHOP, 2006) e (Rabiner, 1989).

Definição 4 (Modelo Oculto de Markov)**.** *Suponha que numa cadeia de observações a n'ésima observação é influenciada por uma variável oculta correspondente. Se as variáveis ocultas são discretas e formam uma cadeia de Markov, diz-se que o modelo é um* ***Modelo***

***Oculto de Markov**. Denotam-se as observações por $O_1, \ldots, O_n$, e as variáreis ocultas por $z_1, \ldots, z_n$.*

Com apenas a condição de que as variáveis ocultas formam uma cadeia de Markov, pode-se provar que a probabilidade $\mathbb{P}\{O_{n+1}|O_1, \ldots, O_n\}$ depende de todos $O_1, \ldots, O_n$, ou seja, a sequência de observações $O = \{O_1, ..., O_n\}$ não é uma cadeia de Markov.

Nesse modelo, as **probabilidades de transição** são dadas por $a_{ij} = \mathbb{P}\{z_n = j|z_{n-1} = i\}$ e as **probabilidades de emissão** são $\mathbb{P}\{O_n|z_n\}$. Caso os valores observados O_n sejam discretos, podendo assumir D valores em $V = \{v_1, \ldots, v_D\}$ e o espaço de estados possua K elementos, ou seja, $S = \{\theta_1, \ldots, \theta_K\}$, as probabilidades de emissão $\boldsymbol{\phi}$ podem ser organizadas em uma matriz KxD, representada por $\boldsymbol{\phi}$. Cada elemento $\boldsymbol{\phi_{kd}}$ equivale à probabilidade de se observar o valor v_d dado que a variável oculta está no estado θ_k. A matriz de transição é construída da mesma maneira que na seção anterior, porém quando se trata de um Modelo Oculto a matriz será representada pela letra $\mathbf{A} = [a_{ij}]_{(i,j)\in S^2}$.

Além disso, a probabilidade marginal $\mathbb{P}\{z_1\}$ que descreve o estado inicial é um vetor $\boldsymbol{\pi}$ de K elementos.

Assim, o modelo λ pode ser descrito por seus três parâmetros: $\lambda = (\boldsymbol{\pi}, \mathbf{A}, \boldsymbol{\phi})$.

2.3.1 Exemplo simples de um Modelo Oculto de Markov

Para que o conceito fique bem claro, um exemplo de um Modelo Oculto de Markov é apresentado a seguir. Considere que o espaço de estados possui dois elementos, $S = \{\theta_1, \theta_2\}$, e a matriz de transição entre os estados ocultos é a mesma do exemplo anterior:

$$\mathbf{A} = \begin{bmatrix} 0.7 & 0.3 \\ 1 & 0 \end{bmatrix} \tag{2.5}$$

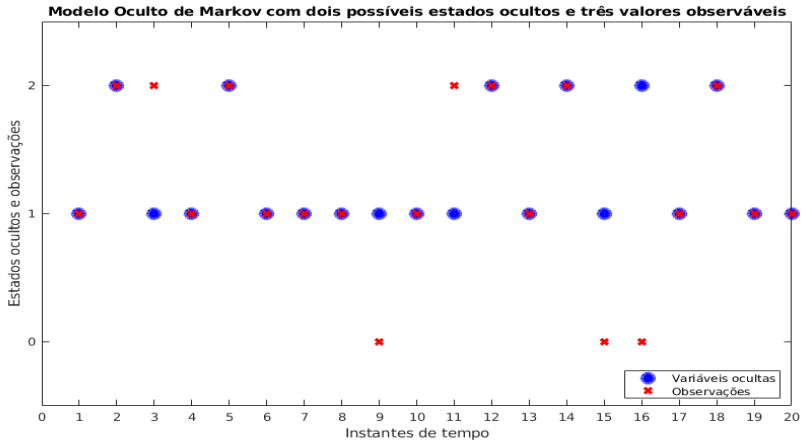
Visto que o sistema inicia no estado θ_1, o vetor do estado inicial é $\boldsymbol{\pi} = \begin{bmatrix} 1 & 0 \end{bmatrix}^T$. Por fim, considere que o sistema pode emitir três valores de forma que $V = \{0, 1, 2\}$. Neste exemplo, em cada instante de tempo existe uma probabilidade maior de que o sistema emita o valor correspondente ao índice do estado oculto, porém existe também a possibilidade de que o sistema emita o valor correspondente ao índice do outro estado ou o valor 0, de acordo com a matriz de emissão:

$$\boldsymbol{\phi} = \begin{bmatrix} 0.2 & 0.7 & 0.1 \\ 0.3 & 0.1 & 0.6 \end{bmatrix} \tag{2.6}$$

Na matriz $\boldsymbol{\phi}$, a primeira coluna corresponde à emissão do valor 0, a segunda ao valor 1 e a terceira ao valor 2. Além disso, a primeira linha corresponde ao estado θ_1 e a segunda linha ao estado θ_2. Logo, por exemplo o elemento $\boldsymbol{\phi}_{12}$ é a probabilidade de emissão do valor 1 dado que a variável oculta se encontra no estado θ_1.

Realizando uma simulação desse sistema em MATLAB, obteve-se o resultado mostrado na Figura 3. Nela, os pontos em azul representam os estados ocultos, ou seja, a trajetória da cadeia de Markov oculta, e as cruzes em vermelho são as observações. Verifica-se, como era esperado de acordo com a matriz da Equação 2.6, que na maior parte dos instantes de tempo o valor observado corresponde ao índice do estado oculto. Isso ocorre nos pontos em que a cruz vermelha e o ponto azul coincidem.

Figura 3: Exemplo de estados ocultos e valores observados de um Modelo Oculto de Markov



Fonte: O Autor (2019)

2.3.2 Problemas básicos relacionados a Modelos Ocultos de Markov

Existem três problemas básicos envolvendo os Modelos Ocultos de Markov (Rabiner, 1989):

Problema 1. Dada uma sequência de observações $O = \{O_1, ..., O_n\}$ e um modelo $\lambda = (\boldsymbol{\pi}, \mathbf{A}, \boldsymbol{\phi})$, como calcular de forma eficiente $\mathbb{P}\{O|\lambda\}$, que é a probabilidade da sequência de observações dado o modelo?

Problema 2. Dada uma sequência de observações $O = \{O_1, ..., O_n\}$ e o modelo $\lambda = (\boldsymbol{\pi}, \mathbf{A}, \boldsymbol{\phi})$, como encontrar a sequência de estados ocultos $Z = \{z_1, ..., z_n\}$ que melhor explica a sequência de observações? Este problema é denominado na literatura como decodificação (*decoding*).

Problema 3. Como ajustar os parâmetros do modelo $\lambda = (\boldsymbol{\pi}, \mathbf{A}, \boldsymbol{\phi})$ para maximizar $\mathbb{P}\{O|\lambda\}$? Em outras palavras, como treinar o modelo?

Neste trabalho, primeiramente um Modelo Oculto de Markov será treinado e em seguida uma sequência de estados ocultos será estimada com base em uma sequência de observações. Logo, os Problemas 3 e 2 deverão ser solucionados.

2.3.3 Solução ao Problema 2

A solução ao problema de encontrar a sequência de estados ocultos que melhor explica a sequência de observações não é única, e depende do significado matemático dado à expressão *explicação ótima*. Duas soluções serão analisadas a seguir:

Pode-se considerar que a cada instante de tempo o estado oculto é aquele que maximiza localmente a probabilidade de se estar no estado θ_i dada a sequência de observações O e o modelo λ. Introduzindo a variável $\gamma_t(i) = \mathbb{P}\{z_t = \theta_i|O, \lambda\}$, esta primeira solução corresponde a maximizar $\gamma_t(i)$ para cada instante de tempo individualmente. Logo, de acordo com esse critério de otimização, os estados ocultos z_t são dados pela Equação 2.7:

$$z_t = \arg\max_{1\leq i\leq K}[\gamma_t(i)], \qquad 0 \leq t \leq n \tag{2.7}$$

Essa solução, apesar de maximizar a esperança do número de estados corretos, apresenta algumas desvantagens. Os estados ocultos são determinados individualmente sem que a sequência deles seja levada em conta. Esse fato pode levar o algoritmo a produzir sequências improváveis ou até mesmo impossíveis quando a matriz $\mathbf{A}$ possui entradas nulas (Rabiner, 1989).

Neste ponto, faz-se necessário ressaltar que a marcha humana pode normalmente ser decomposta em uma sequência de estados cíclica de forma que a probabilidade de realizá-la no sentido inverso pode ser considerada nula. Assim, o algoritmo utilizado para resolver o Problema 2 deve ser capaz de produzir apenas sequências com probabilidade não nula.

Uma segunda solução que contorna essa desvantagem é algoritmo de Viterbi, cujas aplicações são diversas, incluindo decodificação de mensagens, reconhecimento de fala e bioinformática. Ele baseia-se na maximização da probabilidade de toda a sequência de estados $Z = \{z_1, ..., z_n\}$, ou seja, na maximização de $\mathbb{P}\{Z|O, \lambda\}$ que é equivalente a $\mathbb{P}\{Z, O|\lambda\}$. Dessa forma, a sequência ótima Z^* é dada pela Equação 2.8:

$$Z^* = \arg\max_{Z}[\mathbb{P}\{Z, O|\lambda\}] \tag{2.8}$$

No trabalho de (MANNINI; SABATINI, 2011), dados da marcha de três indivíduos foram coletados e os estados ocultos de uma cadeia de Markov foram identificados com um pós-processamento utilizando-se o algoritmo de Viterbi. É importante observar que o cálculo de Z^* através da Equação 2.8 leva em consideração que toda sequência de observações está disponível, o que não é viável em aplicações em tempo real.

Várias propostas foram feitas para contornar essa limitação. Por exemplo pode-se considerar apenas uma janela de tamanho fixo em que apenas uma quantidade limitada de estados é processada (MANNINI; GENOVESE; SABATINI, 2013).

Outra solução é apresentada em (MANNINI; GENOVESE; SABATINI, 2013). Os autores utilizaram um algoritmo de Viterbi de curta duração, em que uma janela variável é considerada para a decodificação dos estados ocultos. Aplicando-se essa técnica por meio de um microprocessador ARM de 32 bits e uma taxa de amostragem de até 500 Hz, a latência de detecção dos estados foi inferior a 100 ms para mais de 75% dos eventos.

Dentro dessas possibilidades, o Algoritmo de Viterbi de curta duração se destaca. Em um primeiro momento ele seria uma boa escolha para resolver o problema da decodificação dos estados em tempo real. Ele está descrito de maneira sucinta abaixo:

2.3.4 Algoritmo de Viterbi de curta duração (*On-line Viterbi*)

Mais de uma versão existe desse algoritmo. Uma versão bastante otimizada foi desenvolvida na seguinte dissertação de mestrado: (ŠRáMEK, 2007). O pseudocódigo, provas e análises de complexidade desse algoritmo podem ser encontradas na dissertação acima e em (ŠRáMEK; BREJOVá; VINAř, 2007). De maneira bastante resumida, o algoritmo consiste em armazenar dinamicamente os possíveis caminhos de estados ocultos e respectivas probabilidades em matrizes dinâmicas. Se em dado instante todos os possíveis caminhos convergem em determinado ponto, todos os estados precedentes ao ponto de convergência são considerados ótimos no critério da Equação 2.8. Para mais detalhes pode-se consultar as referências supracitadas.

Visto que nas buscas realizadas não foi encontrada nenhuma implementação na linguagem C disponível na rede desse algoritmo, o Autor deste Trabalho realizou essa implementação, que está disponível em código aberto no seguinte link: <https://github.com/franciscoambrosio/hidden-markov>. Entretanto, testando-se a implementação do Autor, constatou-se que o programa parava de funcionar depois de um certo tempo em operação e era necessário recomeçá-lo. Provavelmente esse comportamento se deve a algum erro de implementação dada a dificuldade de se trabalhar com alocação dinâmica de memória na linguagem C.

Devido a essas dificuldades, optou-se neste trabalho por utilizar a solução mais simples para o problema da decodificação em tempo real dos estados: uma janela de tamanho fixo em que apenas uma quantidade limitada de estados é processada com o algoritmo de Viterbi convencional.

2.3.5 Solução ao Problema 3

Este problema consiste em ajustar os parâmetros $\lambda = (\boldsymbol{\pi}, \mathbf{A}, \boldsymbol{\phi})$ de forma a maximizar a probabilidade da observação dado o modelo. Não existe uma fórmula analítica que maximiza $\mathbb{P}\{O|\lambda\}$, entretanto é possível encontrar um máximo local de $\mathbb{P}\{O|\lambda\}$ utilizando-se algoritmos iterativos como Baum-Welch, que é um estimador de máxima verossimilhança, ou um método de gradientes (Rabiner, 1989), (MANNINI; SABATINI, 2011). Além disso,

no tocante exclusivamente à matriz $\mathbf{A}$, vale ressaltar que na literatura existem diversas formas de se determinar uma matriz de transição, como (JILKOV; LI, 2004), (ORGUNER; DEMIREKLER, 2008) e (WANG, 2010), que discutem modelos gerais.

Quando se possui um conjunto de dados em que os estados correspondentes às observações já estão identificados, uma boa estimativa inicial para os parâmetros do modelo corresponde a uma contagem do número de vezes em que o sistema esteve em cada estado e dos saltos que realizou. Assim, as probabilidades da matriz $\mathbf{A}$ podem ser estimadas da mesma forma como as entradas da matriz $\mathbf{P}_{empírica}$ foram calculadas em 2.4. Logo, a probabilidade de transição do estado θ_i para o estado θ_j é dada como segue:

$$a_{ij} = p_{ij} = \frac{\text{número de saltos do estado } \theta_i \text{ para o estado } \theta_j}{\text{número de vezes em que o sistema esteve no estado } \theta_i} \tag{2.9}$$

O vetor de estado inicial $\boldsymbol{\pi}$ é estimado por:

$$\pi_i = \frac{\text{número de vezes em que o sistema esteve no estado } \theta_i}{\text{número total de observações}} \tag{2.10}$$

O parâmetro restante $\boldsymbol{\phi}$ depende de como as emissões são definidas. Pode-se modelar as emissões seguindo uma variável discreta assumindo valores em $V = \{v_1, \ldots, v_D\}$, ou contínua. Visto que as saídas dos sensores inerciais, descritos na seção 3.1.1, possuem um passo de discretização muito pequeno comparado ao número de estados, é coerente aproximá-las para variáveis contínuas. Logo, as probabilidades de emissão $\boldsymbol{\phi} = \begin{bmatrix} \phi_1 & \phi_2 & \ldots & \phi_K \end{bmatrix}^T$ são funções densidade de probabilidade. Cada elemento ϕ_i, $1 \leq i \leq K$ é uma função densidade de probabilidade cuja variável aleatória é a observação nos instantes em que $z_t = \theta_i$. Em (Rabiner, 1989) é explicitada a formulação mais geral para as densidades de probabilidade ϕ_i. Neste projeto será considerado o caso particular em que cada ϕ_i é uma distribuição normal de média μ_i e desvio padrão σ_i^2.

$$\phi_i \sim \mathcal{N}(z_t|\mu_i,\, \sigma_i^2, z_t = \theta_i), \qquad 0 \leq i \leq K \tag{2.11}$$

Em outras palavras, esse modelo representa a ideia de que para cada estado oculto, a emissão seguirá uma densidade de probabilidade gaussiana.

A partir de uma sequência de observações é possível estimar os parâmetros μ_i e σ_i^2 , $1 \leq i \leq K$:

$$\mu_i = \frac{1}{N_i} \sum_{t=1}^{n} O_t \Bigg|_{z_t=\theta_i} \tag{2.12}$$

$$\sigma_i = \sqrt{\frac{1}{N_i - 1} \sum_{t=1}^{n} (O_t - \mu_i)^2 \Bigg|_{z_t=\theta_i}} \tag{2.13}$$

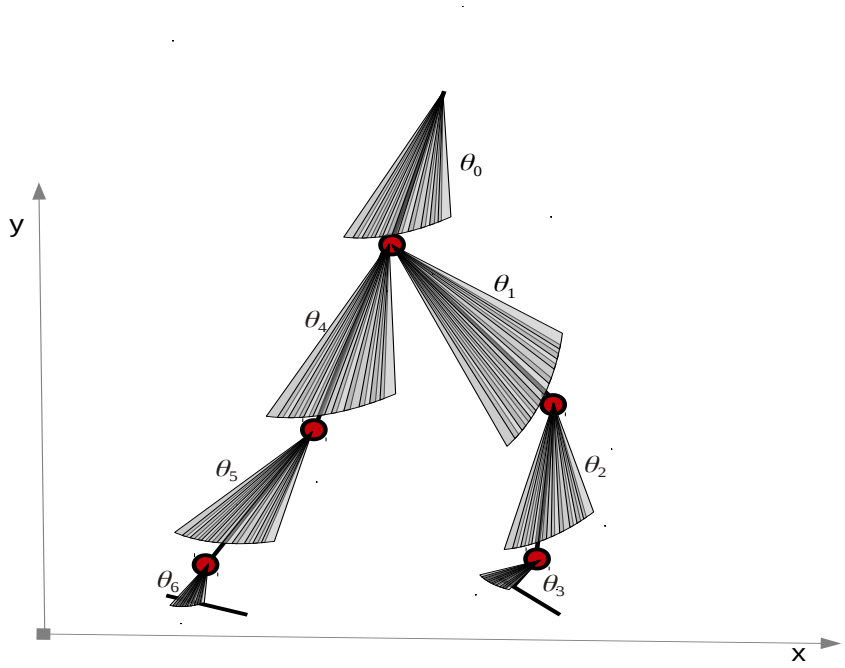
Nas Equações 2.12 e 2.13, n representa o número total de observações e N_i o número de vezes em que o sistema esteve no estado θ_i.

2.4 Modelagem Markoviana do caminhar humano

A marcha humana pode ser segmentada de diversas formas. Algumas não levam em conta nenhum significado físico dos estados, sendo apenas uma subdivisão matemática de variáveis de interesse. Um exemplo dessa abordagem é apresentado abaixo.

2.4.1 Articulações divididas em setores

Figura 4: Articulações divididas em setores



Fonte: Christoph M. Mitschka (2015)

No projeto de iniciação científica **Derivação Empírica de uma Matriz de Probabilidades de Transição do Caminhar Humano com Unidades de Medidas Inerciais** realizado pelo autor de 2015 a 2017 (processo FAPESP 2015/20644-5), uma MPT foi derivada com base em estados Markovianos definidos a partir dos ângulos absolutos descritos pelas juntas de acordo com o modelo utilizado no exoesqueleto Exo-Kanguera, utilizado em tratamentos de reabilitação motora.

O modelo é apresentado nesta seção em linhas gerais. Detalhes técnicos e as equações dinâmicas podem ser encontradas nos relatórios do projeto de doutorado de Christoph

M. Mitschka (processo FAPESP 2012/14074-3), em que foi derivado um controlador Markoviano para esse exoesqueleto.

O exoesqueleto possui três articulações: quadril, joelho e tornozelo, que por convenção serão chamadas de quadril, fêmur e tíbia, respectivamente. As áreas de movimento de cada articulação podem divididas em setores. Um exemplo de divisão é mostrado na Figura 4. Assim, para cada instante de tempo, a configuração espacial do exoesqueleto pode ser aproximada pela combinação dos setores em que se encontra cada articulação.

Cada uma dessas combinações de setores é tratada como um estado Markoviano do exoesqueleto. Uma vez que o exoesqueleto está acoplado ao corpo humano, como mostra a Figura 10, pode-se relacionar diretamente o estado do exoesqueleto com o estado do caminhar da pessoa que o utiliza. Neste caso, cada articulação é dividida em três setores, o que totaliza $3^3 = 27$ estados.

2.4.2 Quatro fases da marcha humana

Da forma como foram definidos os estados na seção acima, os combinações de ângulos das juntas não possuem sentido físico facilmente interpretável. Assim, as fases que serão descritas abaixo foram escolhidas possuindo sentido físico. Existem diversas propostas para isso. Por exemplo (RUETERBORIES et al., 2010) sugere uma divisão em oito fases. Aqui será utilizada uma divisão em quatro fases, como em (MANNINI; SABATINI, 2011), descrita abaixo:

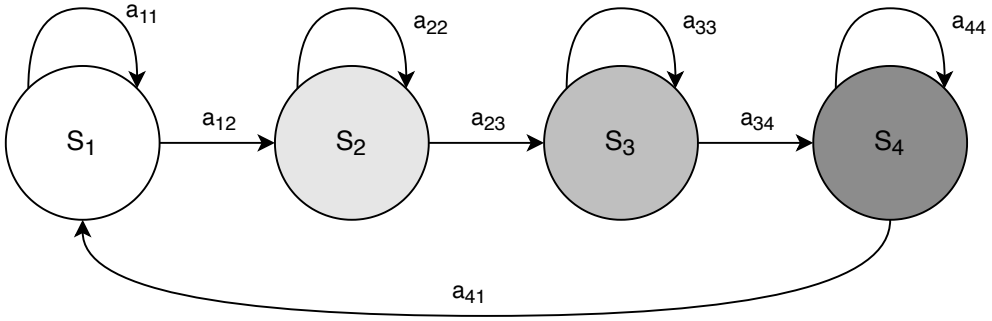
Considere primeiramente os quatro eventos que ocorrem em um ciclo completo da marcha humana convencional:

- **HS *(Heel-Strike)***: Momento em que o pé atinge o solo com o calcanhar
- **FF *(Flat-Foot)***: Momento em que o pé fica paralelo ao solo e em está em contato com o solo
- **HO *(Heel-Off)***: Momento em que o calcanhar sai do solo
- **TO *(Toe-Off)***: Momento em que os dedos saem do solo

Os quatro estados são definidos como os intervalos entre os eventos. O estado S_1 corresponde à fase delimitada por HS e FF. Analogamente, temos S_2: FF-HO, S_3: HO-TO e S_4: TO-HS.

Para modelar o caráter cíclico e unidirecional da marcha humana convencional, é apropriado utilizar um Modelo Oculto de Markov do tipo *left-right*. Nele, o estado só pode pular para o seguinte no ciclo ou para ele mesmo. A Figura 5 mostra o diagrama de estados dessa estrutura.

Figura 5: Diagrama de estados do Modelo Oculto de Markov do tipo *left-right*



Fonte: (MANNINI; GENOVESE; SABATINI, 2013)

3 EXPERIMENTOS

Este capítulo está separado em duas partes. A primeira refere-se aos experimentos realizados pelo autor com o intuito de derivar uma MPT do caminhar de acordo com o modelo das articulações divididas em setores (2.4.1). Já a segunda parte apresenta os experimentos realizados para a derivação do modelo das quatro fases (2.4.2), tanto no treinamento quanto na validação.

3.1 Derivação de uma MPT com base no modelo das articulações divididas em setores

3.1.1 Sensores utilizados

Visto que serão utilizadas IMUs para captar os sinais de interesse no caminhar humano, uma descrição desses sensores é dada a seguir. Normalmente, um sensor inercial é composto por alguns acelerômetros e giroscópios dispostos em diferentes direções, que determinam os graus de liberdade suportados. Encontram-se na literatura muitos trabalhos relacionados à melhoria e ao barateamento desses sensores, como discute (SILVA, 2013), segundo o qual houve um grande progresso nos últimos cinquenta anos.

Uma Unidade de Medida Inercial em estado da arte consiste em um acelerômetro, um sensor magnético e um giroscópio, sendo os três componentes triaxiais. Os sinais dos dois primeiros componentes não apresentam deriva ao longo do tempo, devido à comparação dos sinais com vetores de referência, como a aceleração gravitacional e o campo magnético da Terra. O mesmo não ocorre com o giroscópio, que, dessa forma, deve ser atualizado frequentemente com dados dos outros dois componentes. As IMUs utilizados neste projeto são fabricados com a tecnologia MEMS (Microelectromechanical systems) de estado sólido (XSENS..., 2013).

O acelerômetro pode fornecer a inclinação do corpo. Já a direção dos movimentos pode ser determinada pela saída do sensor magnético. Além disso, pode-se determinar a orientação espacial a partir das condições iniciais e da saída do giroscópio ao longo do tempo (SABATINI, 2011).

O modelo geral para a saída de um sensor inercial após a calibração, de acordo com (INOUE, 2012), pode ser dado por:

$$y_m = (1 + k)y_t + b(t) \tag{3.1}$$

sendo y_m a saída apresentada pelo sensor, y_t o valor exato da medida, k o erro de fator de escala e $b(t)$ a polarização. Considera-se, geralmente, que o erro de fator de escala é randômico. O software de fábrica que acompanha os sensores implementa um filtro de

Kalman que fusiona as saídas inerciais com os dados do magnetômetro nas três dimensões para estimar a orientação das IMUs de maneira ótima (XSENS. . . , 2013).

As medidas para o modelo das articulações divididas em setores foram realizadas com Unidades de Medidas Inerciais MTw Awinda (Figura 6) do MTw Development Kit da XSens que foi financiado pelo CNPq. O kit contém sensores inerciais sem fio que podem gravar o posicionamento dos membros de forma precisa.

Figura 6: Unidade de medida inercial MTw Awinda

Fonte: (XSENS. . . , 2013)

O kit também conta com um software que permite visualizar os dados, trabalhar com eles e sincronizar os sensores nos experimentos. Isso possibilita uma análise completa do caminhar humano. Na Figura 7 é mostrado um exemplo de utilização do software em que algumas saídas das IMUs são representadas graficamente.

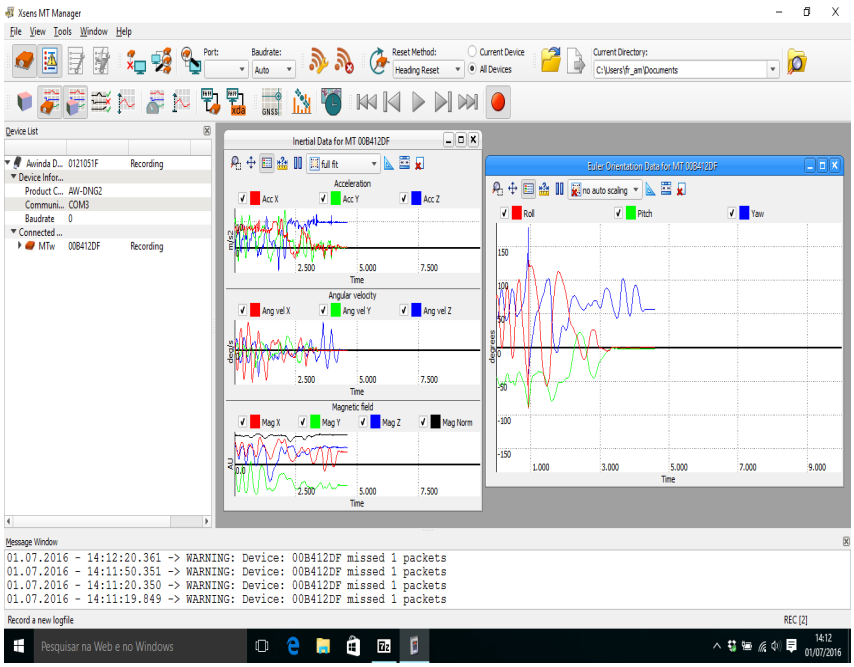
3.1.2 Realização

Para que se medissem os ângulos absolutos referentes a cada uma das juntas explicitadas em (2.4.1), três IMUs foram acopladas ao corpo humano como mostra a Figura 8. Com essa configuração, foram coletados dados em diversos ambientes reais de caminhada: planos horizontais e inclinados, escadas e superfícies irregulares.

Além disso, foram realizados experimentos em locais de transição entre esses ambientes. Em especial, a transição entre plano horizontal e escada foi tratada como uma perturbação na trajetória do caminhar na elaboração do artigo *Recursive Linear Quadratic Regulator Subject to Markovian Jump Linear Systems in a Robotic Application System for Rehabilitation* (MITSCHKA et al., 2016). Nele, foram utilizadas as MPTs derivadas aqui. Trata-se, portanto, da primeira contribuição do autor em um publicação.

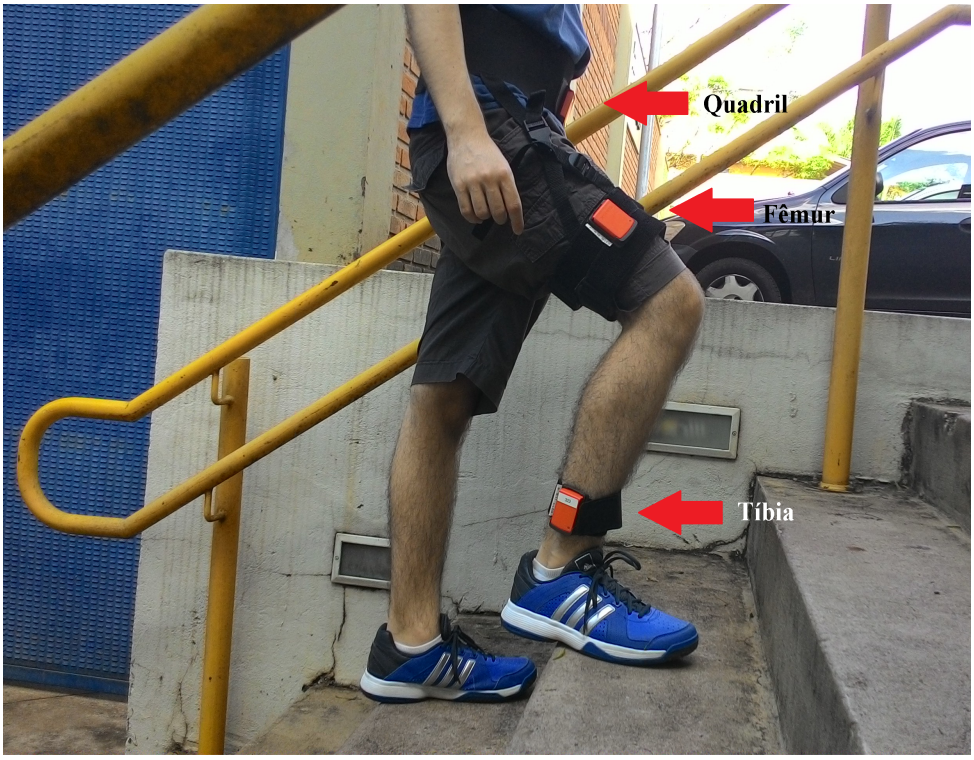
Visto que o método de derivação da matriz é estatístico, coletou-se um volume grande de dados, de forma que os resultados pudessem ser confiáveis. Por exemplo, no

Figura 7: Software MTw Manager



Fonte: O Autor (2020)

Figura 8: IMUs acopladas ao corpo humano para a aquisição dos dados



Fonte: O Autor (2016)

ambiente de escada, realizaram-se cerca de 1500 passos (750 de subida e 750 de descida).

Nesse contexto, faz-se necessário relatar as principais dificuldades enfrentadas no decorrer dos experimentos. Primeiramente, no decorrer desse período, o laboratório adquiriu sensores inerciais da geração mais recente. Assim, foi necessário contactar o fabricante para que se soubesse como sincronizar as duas gerações de IMUs em uma mesma base receptora dos sinais. O problema foi resolvido com uma atualização de software.

Além disso, pelo fato de os sensores serem sem fio, alguns conjuntos de dados tiveram que ser descartados devido a interrupções da comunicação entre os sensores e a base receptora. Não obstante, vale ressaltar a dificuldade em manter os níveis das baterias próximos, para que nenhum sensor ficasse sem energia durante os experimentos. As baterias, por sua vez, demoram algumas horas para carregarem. Dessa forma, os experimentos deviam ser planejados e preparados com antecedência.

3.2 Treinamento do Modelo Oculto de Markov do tipo *left-right* com quatro fases

3.2.1 Sensores utilizados

Devido às condições restritivas decorrentes da pandemia de SARS-CoV-2, não foi possível ter acesso às IMUs do laboratório no período em que foram realizados os experimentos para este segundo modelo. Dessa forma, procurou-se uma solução alternativa que estivesse ao alcance do autor. A solução encontrada foi utilizar os dados da IMU presente em um *smartphone* de modelo Moto G5, da marca Motorola, cujo sistema operacional é o Android 8.1.0. Nos documentos fornecidos pelo fabricante não consta informação sobre o modelo exato da IMU existente dentro do aparelho, porém alguns estudos indicam que a performance não é muito inferior a IMUs profissionais (PFEAU; WELLE, 2015), (Zhi; Xu; Schwertfeger, 2019).

3.2.2 Escolha das saídas de interesse

Para que o modelo da Figura 5 seja aplicado, duas escolhas precisam ser feitas. Primeiramente, deve-se escolher como os estados ocultos serão identificados na fase de treinamento. Trata-se do etiquetamento dos estados (do inglês *labeling*). Cada variável oculta $z_1, \ldots, z_n$ deve receber um valor em em $S = \{S_1, \ldots, S_K\}$. Aqui $K = 4$ pois temos 4 estados possíveis. Na etapa de treinamento o *labeling* deve possuir uma grande acurácia. Pode-se utilizar para isso por exemplo sensores de pressão, câmeras, sistemas de rastreamento ou variáveis físicas cinemáticas ou dinâmicas. Em muitos casos o etiquetamento é realizado por um especialista na área que classifica *frames* em um vídeo. Outra possibilidade de implementação mais simples é utilizar um conjunto de regras que as saídas de interesse devem satisfazer em cada evento.

A segunda escolha que deve ser feita é qual ou quais variáveis serão consideradas

como observações $O = \{O_1, ..., O_n\}$. Em geral utilizam-se para isso variáveis físicas cinemáticas ou dinâmicas. Assim, as observações são em geral um vetor de uma ou mais saídas específicas de sensores, como acelerômetros e giroscópios.

Neste projeto uma IMU presente em um *smartphone* foi fixada na parte superior do pé, e as observações foram consideradas as saídas do giroscópio na direção médio-lateral. Sabe-se que o perfil da curva de velocidade angular do pé na direção médio-lateral é uma sequência bastante estável de arcos e planos (MANNINI; SABATINI, 2011). Tal fato é essencial para a detecção dos eventos, pois eles podem ser identificados como pontos na curva que satisfazem um determinado conjunto de regras. Optou-se por realizar o etiquetamento dos estados utilizando-se regras baseadas nas regras definidas em (MANNINI; SABATINI, 2011), que determinam em que momentos ocorrem os eventos HS, FF, HO e TO. As regras do trabalho de (MANNINI; SABATINI, 2011) são mostradas a seguir:

- t_{HS}: Ocorre logo antes do pico negativo da velocidade angular. Define-se como o instante em que a diferença absoluta entre o sinal filtrado com o filtro passa-baixas e o sinal não filtrado é máxima.
- t_{FF}: Instante em que a velocidade angular fica maior que um threshold de -50 °/s, estando previamente menor que -50 °/s.
- t_{HO}: Instante em que a velocidade angular volta a ficar menor que um threshold de -50 °/s.
- t_{TO}: Instante em que a velocidade angular se anula, estando previamente negativa.

Essas regras foram adaptadas e foram inseridas regras adicionais para o processamento não causal dos dados. Seja $\omega(n)$, $n \in \mathbb{N}_0$, a n'ésima discretização da velocidade angular do pé na direção médio-lateral e seja $\tilde{\omega}(n)$ a velocidade angular $\omega(n)$ filtrada por um filtro Butterworth de segunda ordem passa-baixas com frequência de corte de 15 Hz. Definimos $\delta(n) = |\omega(n) - \tilde{\omega}(n)|$.

Além disso, seja $P_{\tilde{\omega}} = \{p_1, ..., p_q\}$ o conjunto de todos os $n \in \mathbb{N}_0$ que satisfazem $\tilde{\omega}(n) = 0$ °/s. A posição angular $\tilde{\theta}(n)$ é dada pela integração numérica de $\tilde{\omega}(n)$ e a aceleração angular $\frac{d\tilde{\omega}}{dt}(n)$ pela derivação numérica. As seguintes regras foram utilizadas:

- $T_{FF} = \{n \in \mathbb{N}_0 \mid \tilde{\omega}(n) \geq -50$ °/s e $\tilde{\omega}(n-1) < -50$ °/s e $\tilde{\theta}(n) > -15$ °$\}$
- $T_{HO} = \{n \in \mathbb{N}_0 \mid \tilde{\omega}(n) \leq -50$ °/s e $\tilde{\omega}(n-1) > -50$ °/s e $\tilde{\theta}(n) < 0$ °$\}$
- $T_{TO} = \{p_k \in P_{\tilde{\omega}} \mid \tilde{\omega}(p_k) = 0$ °/s e $\frac{d\tilde{\omega}}{dt}(p_k) > 2000$ °/s² e $\frac{d\tilde{\omega}}{dt}(p_{k+1}) < -1500$°/s²$\}$
- Finalmente, para cada $t_{TO} \in T_{TO}$, seja p_{k+} o $p_k \in P_{\tilde{\omega}}$ mais próximo de t_{TO} tal que $p_k > t_{TO}$. Seja t_{FF+} o $t_{FF} \in T_{FF}$ mais próximo de p_{k+} tal que $t_{FF} > p_{k+}$. Faça $t_{HS} = \text{argmax}_n\, \delta(n), p_{k+} < n < t_{FF+}$. Temos que T_{HS} é o conjunto de todos t_{HS}.

3.2.3 Realização

Figura 9: *smartphone* acoplado ao pé

Fonte: O Autor (2020)

O *smartphone* foi acoplado ao pé na parte superior (Figura 9). Para a transmissão dos dados em tempo real para um laptop da marca Dell, modelo Inspiron 15 com um processador Intel Core i5 de oitava geração, foi utilizado o aplicativo Sensor Node Free. Para a recepção, gravação para pós-processamento e processamento desses dados em tempo real, foi utilizado o código em Python disponível no Apêndice B. Esses dados foram coletados com um período de amostragem de 6 ms, o que corresponde a 166,67 Hz.

Inicialmente, realizaram-se duas sessões de gravação dos dados para pós-processamento: uma foi utilizada para treinamento e outra para validação do modelo. Ambas foram de curta duração e realizadas em um corredor, visto que não foi possível ter acesso à esteira do laboratório, dadas as condições de isolamento social.

Nesse sentido, foram utilizados apenas 6 passos completos para o treinamento do modelo. Os dados da saída do giroscópio foram filtrados com um filtro Butterworth passa-baixas de segunda ordem com frequência de corte de 15 Hz apenas para o etiquetamento dos estados. Já a matriz de emissão foi calculada utilizando os dados sem filtragem. O treinamento foi realizado em Matlab utilizando-se os estados definidos em 2.4.2, as regras 3.2.2 e as equações 2.9, 2.12 e 2.13. O vetor de estado inicial $\boldsymbol{\pi}$ foi considerado uma distribuição uniforme entre os 4 estados possíveis.

Aqui, vale ressaltar que o treinamento foi realizado de forma não causal, ou seja, os dados não foram processadas em sequência em um único *loop*. Logo, foi necessário adicionar

algumas condições extras para retirar pontos que satisfaziam as regras apresentadas em (MANNINI; SABATINI, 2011) mas estavam em locais errados da sequência de estados. As regras adaptadas foram as apresentadas em 3.2.2. Por exemplo, a velocidade angular passa de um valor inferior a -50 °/s para um valor maior que -50 °/s duas vezes em um ciclo completo da marcha (Figura 12). Entretanto, só a primeira vez deve ser considerada, pois na segunda vez essa regra não faz mais sentido, já que a marcha está em outro estado e outra regra deve ser verificada para caracterizar o evento correspondente.

Outro exemplo é que a velocidade angular possivelmente se anula diversas vezes durante um ciclo, em especial na fase de *flat-foot*. Entretanto, só deve ser considerado o ponto que ocorre ao final da fase S_3.

Assim, a experiência do autor adquirida com o presente trabalho sugere que um processamento causal é de mais simples implementação.

A validação do modelo foi realizada com o segundo conjunto de dados aplicando-se o algoritmo de Viterbi convencional em pós-processamento. Portanto, toda a sequência de observações está disponível, o que não ocorre em aplicações em tempo real. Para isso utilizou-se a biblioteca em Matlab (MURPHY, 1998). Esse procedimento foi realizado com os dados não filtrados e com os dados filtrados. Os resultados foram comparados.

Por fim, realizou-se uma nova sessão de gravação em que os estados Markovianos foram estimados em tempo real utilizando-se o algoritmo de Viterbi em uma janela de tamanho fixo deslizante. Adotou-se uma janela de duração igual a 15 períodos de amostragem, o que corresponde a 90 ms. Os dados dentro dessa janela foram filtrados em tempo real por um Butterworth filtro passa-baixas de segunda ordem com frequência de corte de 15 Hz. Logo em seguida aplicou-se o algoritmo de Viterbi dentro da janela. Para isso utilizou-se a biblioteca *hmmlearn* em Python. A cada iteração considerou-se o estado inicial como o a saída da iteração anterior.

A saída a cada iteração foi considerada como o segundo elemento do vetor de estados ocultos estimados pelo algoritmo de Viterbi, já que o primeiro elemento sempre correspondia à saída da iteração anterior. O código completo em Python dos experimentos está disponível no Apêndice B. Nessa sessão foram gravados dados de 17 passos em um corredor. Esses dados também foram aproveitados para validar novamente o algoritmo em pós-processamento, de forma a aumentar a significância estatística.

Finalmente, foi demonstrada uma aplicação que consiste em medir a duração dos passos bem como a duração das fases. Observa-se que em padrões de marcha com alguma debilidade frequentemente a duração e proporções entre as durações das fases difere em relação ao padrão de marcha normal. Aqui foi considerada a fase denominada de *Stance*, que corresponde ao período em que o pé está em contato com o solo, ou seja, trata-se da união das fases S_1, S_2 e S_3 definidas em 2.4.2.

4 RESULTADOS

Assim como nos experimentos, os resultados serão divididos em duas partes, sendo a primeira corresponde ao modelo das articulações divididas em setores, e a segunda ao Modelo Oculto de Markov com quatro estados.

4.1 Articulações divididas em setores

Para cada cenário de caminhada, bem como para cada experimento realizado, a MPT encontrada será diferente. Entretanto, a matriz referente a indivíduos sem dificuldades para caminhar e de estatura média não apresentaram grandes desvios. Escolhendo-se o ambiente da transição entre plano horizontal e escada, a MPT encontrada neste caso é mostrada na Figura 10.

Observe que as maiores probabilidades concentram-se nas diagonais principal e nas diagonais próximas a ela. Isso se deve ao fato de que em intervalos de tempo pequenos, a Cadeia de Markov apresenta estados sequenciais repetidos, uma vez que a taxa de amostragem é alta.

Figura 10: MPT do caminhar humano referente à transição entre plano horizontal e escada

P

0.4384	0.1370	0.2740	0	0	0	0.0137	0	0.0137	0.0411	0.0137	0.0548	0	0	0.0137	0	0	0	0	0	0	0	0	0	0	0	0	0
0.1304	0.5304	0.1304	0.0087	0.0174	0	0.0087	0.0261	0	0.0174	0.0783	0.0348	0	0	0.0087	0	0.0087	0	0	0	0	0	0	0	0	0	0	0
0.0535	0.0736	0.6722	0	0	0.0401	0.0033	0	0.0334	0.0100	0.0067	0.0970	0	0	0.0100	0	0	0	0	0	0	0	0	0	0	0	0	0
0.0833	0	0.0833	0.2500	0.0833	0.1667	0	0.2500	0	0	0	0	0.0833	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.0417	0.1250	0	0.0833	0.2917	0.1667	0	0.0417	0	0	0	0	0	0.0833	0	0	0.1250	0.0417	0	0	0	0	0	0	0	0	0	0
0	0	0.0616	0.0137	0.0342	0.6370	0	0	0.1301	0	0	0.0205	0	0	0.0890	0	0	0.0137	0	0	0	0	0	0	0	0	0	0
0.0159	0	0	0.0317	0	0	0.3810	0.1587	0.2698	0	0	0	0.0159	0	0.0159	0.0952	0	0.0159	0	0	0	0	0	0	0	0	0	0
0.0211	0.0211	0	0	0.0211	0	0.0105	0.1579	0.4211	0.2211	0	0	0	0	0	0	0.0316	0.0737	0.0211	0	0	0	0	0	0	0	0	0
0.0020	0.0060	0.0120	0	0.0040	0.0339	0.0279	0.0459	0.7106	0	0	0	0	0	0.0020	0.0040	0.0160	0.1357	0	0	0	0	0	0	0	0	0	0
0.0043	0.0043	0	0	0	0	0	0	0	0.4353	0.1810	0.2026	0.0302	0.0216	0.0129	0.0388	0.0216	0.0172	0.0043	0.0172	0.0043	0	0	0	0.0043	0	0	
0	0.0152	0.0101	0	0.0025	0	0	0	0	0.0886	0.5772	0.1696	0.0051	0.0380	0.0051	0	0.0506	0.0051	0.0076	0.0203	0.0025	0	0	0	0.0025	0	0	
0	0.0003	0.0111	0	0	0	0	0	0.0003	0.0138	0.0159	0.8564	0.0012	0.0006	0.0312	0.0021	0.0003	0.0453	0.0003	0.0009	0.0186	0	0	0.0006	0	0	0.0015	
0	0	0	0.0106	0	0	0	0	0.0106	0.1277	0.0106	0.0319	0.3830	0.0851	0.1170	0.0957	0.0213	0.0213	0	0	0	0.0532	0	0.0106	0.0106	0.0106	0	
0	0.0029	0	0	0.0088	0.0029	0	0	0	0.0117	0.0381	0.0117	0.0176	0.4780	0.2522	0.0147	0.0499	0.0557	0	0	0	0	0.0205	.0205	0.0029	0.0088	0.0029	
0	0	0	0	0.0004	0.0057	0	0	0.0009	0.0013	0.0013	0.0471	0.0065	0.0392	0.7529	0.0022	0.0074	0.0924	0	0	0.0004	0.0009	0.0022	0.0336	0	0.0004	0.0052	
0.0019	0	0	0	0	0	0.0057	0.0038	0.0019	0.0230	0.0172	0.0077	0.0211	0.0077	0.0211	0.0077	0.0096	0.4636	0.1456	0.2356	0.0019	0	0	0.0230	0.0096	0	0.0153	
0	0.0005	0	0	0	0	0	0.0042	0.0037	0.0011	0.0100	0	0.0021	0.0148	0.0106	0.0476	0.5531	0.3051	0	0.0011	0	0	0.0021	0.0005	0.0026	0.0243	0.0143	
0	0	0.0005	0	0	0.0008	0.0003	0	0.0048	0.0002	0.0003	0.0103	0.0002	0.0011	0.0193	0.0093	0.0483	0.8727	0	0	0.0003	0	0.0008	0.0004	0.0008	0.0025	0.0295	
0	0	0	0	0	0	0	0	0	0.0784	0.0392	0.0392	0	0	0	0	0.0196	0	0.4510	0.0588	0.1961	0	0	0.0392	0.0392	0.0196	0.0196	
0	0.0833	0.0208	0	0	0	0	0	0	0	0.2083	0.0417	0.0208	0	0	0	0	0	0.1042	0.3958	0.0833	0	0	0	0.0417	0		
0.0061	0	0.0121	0	0	0	0	0	0	0.0030	0	0.1758	0	0	0.0030	0	0.0030	0.0212	0.0242	0.0121	0.6667	0	0	0.0242	0.0061	0	0.0424	
0	0	0	0	0	0	0	0	0	0	0	0.0345	0.1034	0	0	0	0	0	0.1724	0	0.0345	0.3448	0.0345	0.1379	0.1034	0	0.0345	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0714	0	0	0.0286	0.0143	0.0143	0.0429	0.0143	0.0571	0.4429	0.1714	0.0286	0.1000	0.0143
0	0	0	0	0	0.0023	0	0	0	0	0	0.0113	0	0.0091	0.1247	0.0023	0	0.0272	0.0023	0	0.0249	0.0045	0.0340	0.6644	0	0.0136	0.0794	
0	0	0	0	0	0	0.0064	0.0064	0	0.0064	0	0	0.0064	0	0	0.0955	0.0064	0.0382	0.0064	0.0064	0	0.0191	0	0.0127	0.5414	0.0637	0.1847	
0	0	0	0	0	0	0	0.0074	0.0037	0	0	0	0	0.0037	0.0074	0.0148	0.1556	0.1222	0.0037	0.0037	0	0	0.0074	0.0111	0.0889	0.3778	0.1926	
0	0	0	0	0	0.0007	0	0	0.0021	0	0	0.0021	0	0	0.0089	0.0041	0.0185	0.2707	0	0	0.0082	0.0014	0.0027	0.0164	0.0117	0.0370	0.6155	

P

Fonte: O Autor (2016)

Figura 11: MPT do camihar humano referente ao plano horizontal

P

0.8842	0.0788	0	0.0025	0	0	0	0	0	0.0025	0	0	0	0	0	0	0	0	0.0320	0	0	0	0	0	0	0	0
0.0001	0.9238	0	0	0.0112	0	0	0.0001	0	0	0.0009	0	0	0	0	0	0	0	0	0.0002	0.0589	0	0	0.0049	0	0	0	0
0	0.1111	0.8889	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.0101	0.0040	0	0.8589	0.0101	0	0.0806	0	0	0	0	0	0.0282	0	0	0.0060	0	0	0	0	0	0.0020	0	0	0	0	0	0
0	0.0030	0	0.0016	0.8603	0.0003	0.0004	0.0791	0	0	0	0	0	0.0177	0.0001	0	0.0017	0	0	0	0	0	0.0355	0	0	0.0003	0
0	0	0	0	0.0405	0.9189	0	0	0	0	0	0	0	0	0.0405	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0.8113	0	0	0	0	0	0	0	0	0.1849	0.0038	0	0	0	0	0	0	0	0	0	0	0
0	0.0004	0	0	0.0017	0	0.0006	0.8129	0	0	0	0	0	0.0002	0	0	0.1828	0	0	0	0	0	0	0	0	0	0.0013	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.1081	0.0189	0	0	0	0	0	0	0	0.8405	0.0297	0	0.0027	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0.0951	0	0	0.0010	0	0	0	0	0.0002	0.8991	0	0	0.0043	0	0	0.0003	0	0	0	0	0	0	0	0	0	0	0
0	0	0.2000	0	0	0	0	0	0	0	0	0.8000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.0002	0	0	0.0055	0.0002	0	0	0	0	0.0124	0.0004	0	0.9518	0.0288	0.0002	0.0002	0	0	0	0	0	0.0002	0	0	0	0	0	0
0	0.0002	0	0	0.0061	0	0	0	0	0	0.0227	0	0.0020	0.9665	0.0009	0	0.0012	0	0	0	0	0	0.0003	0	0	0	0	0
0	0	0	0	0	0.0027	0	0	0	0	0	0.0009	0.0027	0.0297	0.9640	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0.0932	0	0	0.8657	0.0094	0	0	0	0	0	0	0	0	0.0317	0	0
0	0	0	0	0	0	0	0.0004	0	0	0.0003	0	0.0009	0.0694	0	0.0047	0.8398	0	0	0	0	0	0.0006	0	0.0001	0.0838	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.6136	0	0	0.3864	0	0	0	0	0	0
0	0.0004	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0007	0.7397	0	0.0004	0.2584	0	0	0.0004	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0.0842	0	0	0.0105	0	0	0	0	0	0.0053	0	0	0	0	0	0	0	0	0.8868	0.0079	0.0026	0.0026	0	0	0
0	0	0	0.0001	0.0421	0	0	0.0049	0	0	0	0	0.0001	0.0104	0	0	0.0009	0	0	0.0001	0	0.0017	0.9109	0	0	0.0289	0	0
0	0	0	0	0	0.5000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.5000	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0750	0	0	0	0	0	0	0	0	0.9012	0.0239	0	0
0	0	0	0	0.0001	0	0	0.0191	0	0	0	0	0	0.0017	0	0.0004	0.0413	0	0	0.0001	0	0	0.0113	0	0.0010	0.9250	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

P

Fonte: O Autor (2016)

Comparando-se as Figuras 10 e 11, verifica-se que a MPT referente à transição entre plano horizontal e escada possui um número maior de entradas não nulas, ou seja, há uma maior diversidade de movimentos realizados. Isso está de acordo com o esperado, já que nesse caso a complexidade do cenário é maior.

4.2 Modelo oculto de Markov do tipo *left-right* com quatro fases

Apesar de não ter sido utilizado um método de elevada acurácia (como sensores de pressão, câmeras e rastreadores de movimento) para o etiquetamento dos estados na fase de treinamento, o método baseado em um conjunto de regras apresentou resultados satisfatórios. Na Figura 12 são mostrados os 4 eventos definidos na seção 2.4.2 que foram detectados com base nas regras. Um ciclo completo compreende os estados S_1, S_2, S_3 e S_4, que em seguida voltam a se repetir no ciclo seguinte.

Observa-se na Figura 12 que o instante t_{HS} (momento em que o calcanhar atinge o solo) está um pouco antes do primeiro pico negativo da velocidade angular. Em seguida o intervalo entre t_{FF} e t_{HO} configura o estado em que todo o pé está em contato com o solo, o que pode ser associado ao valor absoluto da velocidade angular próximo de zero. Em seguida, entre os eventos t_{HO} e t_{TO} encontra-se o maior pico negativo da velocidade angular, em que o calcanhar sai do solo e a ponta do pé ainda está em contato com o solo. Por fim, após os dedos saírem do solo em t_{TO}, a velocidade angular muda para o lado positivo, visto que a rotação do pé muda de sentido.

Este ciclo de dois picos negativos seguidos de um pico positivo possibilita também a aplicação do algoritmo de (LEE; PARK, 2011) para comparação, mesmo que ele tenha sido inicialmente projetado para dados de um sensor afixado na canela.

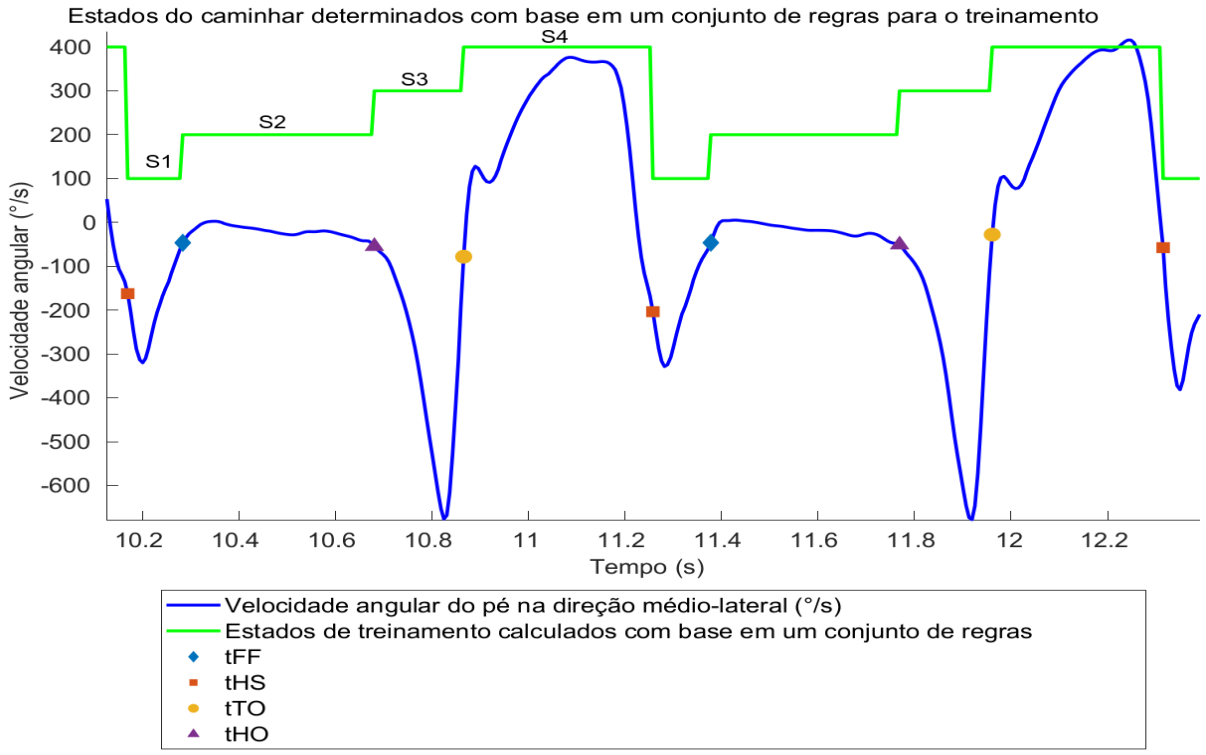
Em seguida os seguintes parâmetros do modelo foram encontrados:

Matriz de Probabilidades de Transição

$$\mathbf{A} = \begin{bmatrix} 0.948 & 0.052 & 0 & 0 \\ 0 & 0.985 & 0.015 & 0 \\ 0 & 0 & 0.970 & 0.030 \\ 0.016 & 0 & 0 & 0.984 \end{bmatrix}$$

Estado (i)	Média das emissões (μ_i) [°/s]	Desvio padrão das emissões (σ_i) [°/s]
1	-199.7	113.8
2	-18.5	14.9
3	-298.2	221.6
4	232.6	163.8

Figura 12: Estados do caminhar determinados com base em um conjunto de regras para treinamento do modelo de Markov



Fonte: O Autor (2020)

Nota: Observa-se a sequência de dois picos negativos seguidos de um pico positivo da velocidade angular na direção médio-lateral de um giroscópio posicionado sobre o pé

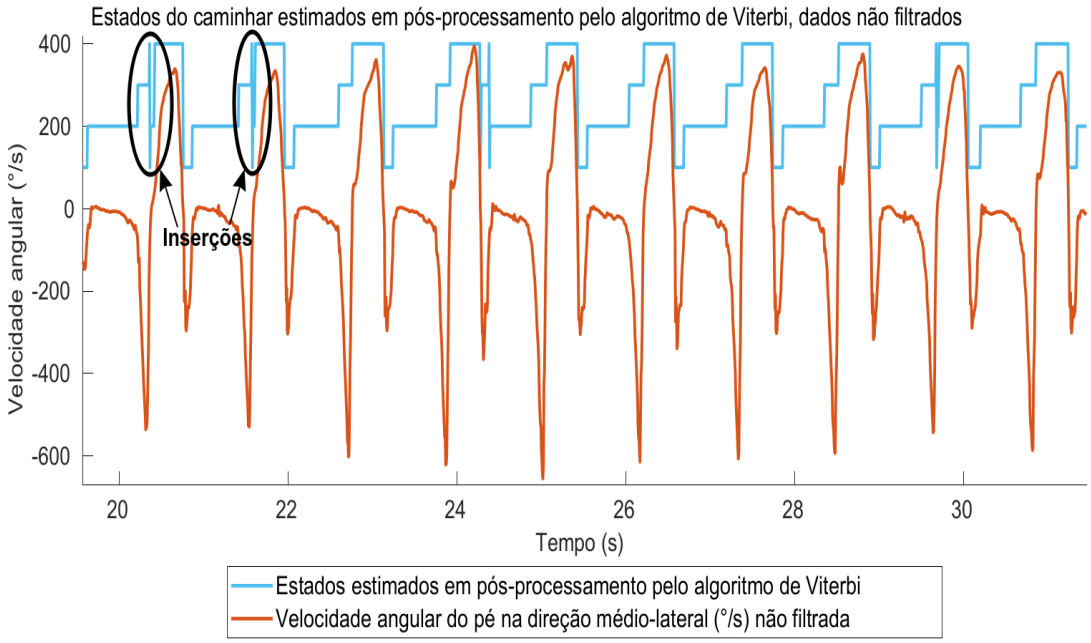
Para a validação do modelo foi utilizado o segundo conjunto de dados, em que haviam dados de dez passos completos. Os estados estimados com o algoritmo de Viterbi com os dados não filtrados são mostrados na Figura 13. Observa-se que houve uma alta taxa de inserções, em que um ciclo completo de curta duração é inserido no meio de um ciclo com duração normal de caminhada. Ocorreram 4 inserções em 10 passos, o que é insatisfatório para aplicações práticas. Já quando se realizou o mesmo procedimento com os dados filtrados com o filtro passa-baixas a taxa de inserções foi bastante reduzida (Figura 14). Ocorreu 1 inserção no mesmo conjunto de dez passos. Esse resultado é bastante satisfatório levando em conta a pequena quantidade de dados de treinamento, o que mostra que em alguns casos é possível realizar o treinamento rapidamente. Visto que posteriormente foram gravados mais 17 passos para o algoritmo em tempo real, testou-se também o pós-processamento nesses dados filtrados e constatou-se 1 inserção nos 17 passos. Isso totaliza 2 inserções em 27 passos.

Além disso, pode-se verificar pela análise visual da Figura 14 que as formas de onda e os estados correspondentes são coerentes de um ponto de vista qualitativo em relação

aos estados definidos na Figura 12.

Entretanto, vale ressaltar que o sistema pode não ser generalizável para outros indivíduos com padrões de caminhada diferentes, uma vez que o modelo foi treinado com apenas um indivíduo.

Figura 13: Estados do caminhar estimados em pós-processamento pelo algoritmo de Viterbi com dados não filtrados

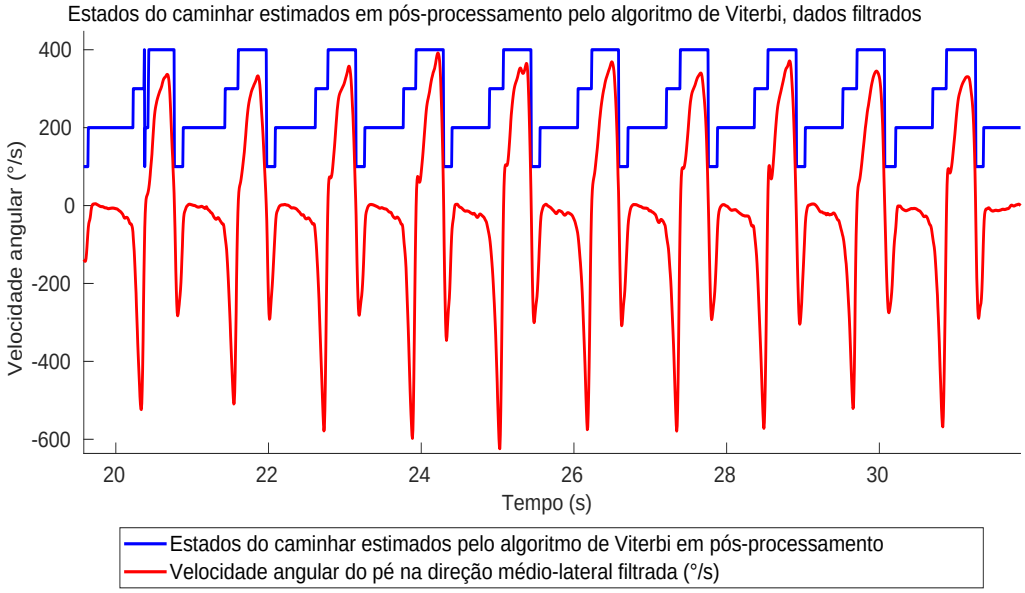


Fonte: O Autor (2020)

Com esse mesmo modelo, porém agora aplicando-se o algoritmo de Viterbi em uma janela deslizante de tamanho fixo para a estimação dos estados em tempo real, foi obtido o resultado da Figura 15. Observa-se que a taxa de inserções é ainda mais elevada do que no caso do pós-processamento com dados não filtrados. Ocorreram 28 inserções em 17 passos. Essas inserções provavelmente inviabilizariam uma aplicação prática em tempo real. Além disso, surge um atraso que está relacionado à duração da janela de tamanho fixo.

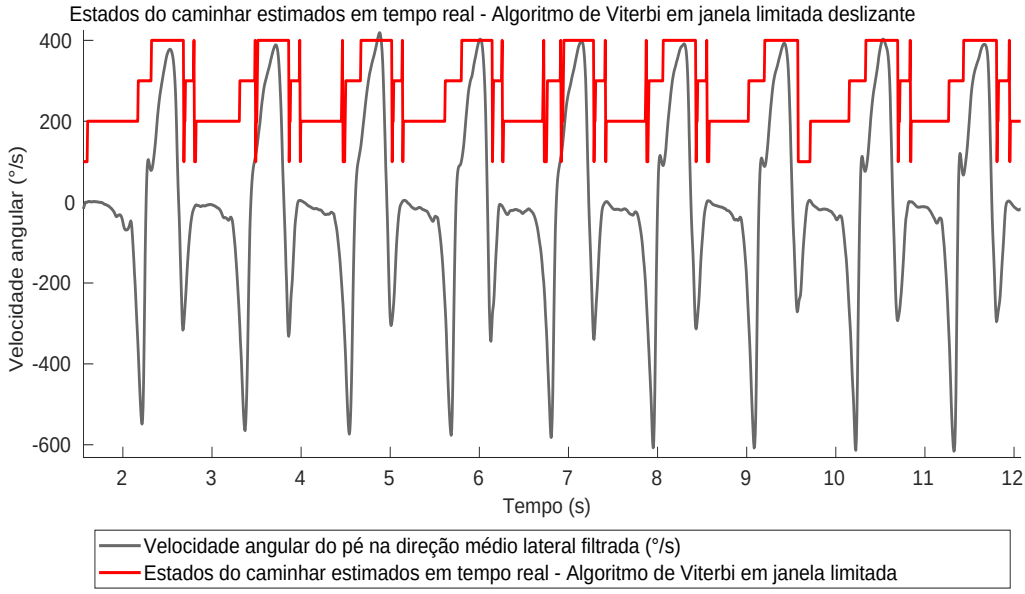
Por fim, as durações dos passos e das fases de *Stance* foram computadas para o algoritmo de Viterbi em pós-processamento com os dados filtrados (caixa azul na Figura 17) e para o algoritmo de Viterbi em uma janela deslizante de tamanho fixo (caixa vermelha na Figura 17). Esses dados são mostrados na Figura 17 na forma de um diagrama de caixa (*boxplot*). Nele são representados a mediana, a média, os quartis inferior e superior, o intervalo que conteria cerca de 99.3% dos dados caso a distribuição fosse normal e os *outliers*.

Figura 14: Estados do caminhar estimados em pós-processamento pelo algoritmo de Viterbi com dados filtrados



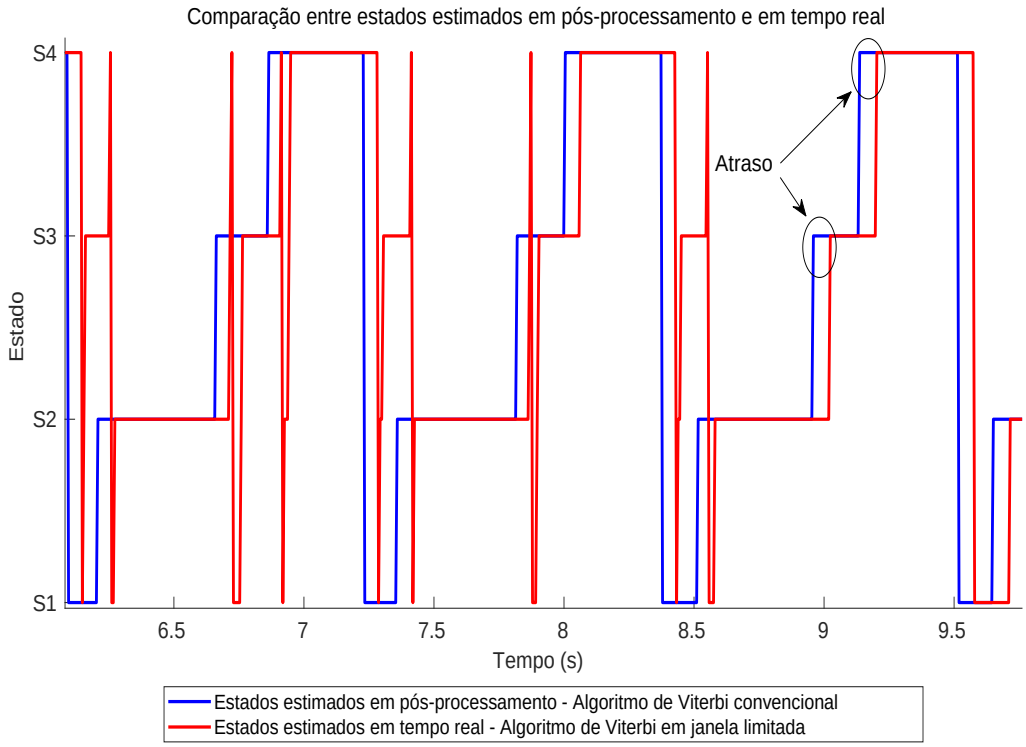
Fonte: O Autor (2020)

Figura 15: Estados do caminhar estimados em tempo real pelo algoritmo de Viterbi em uma janela limitada de tamanho fixo



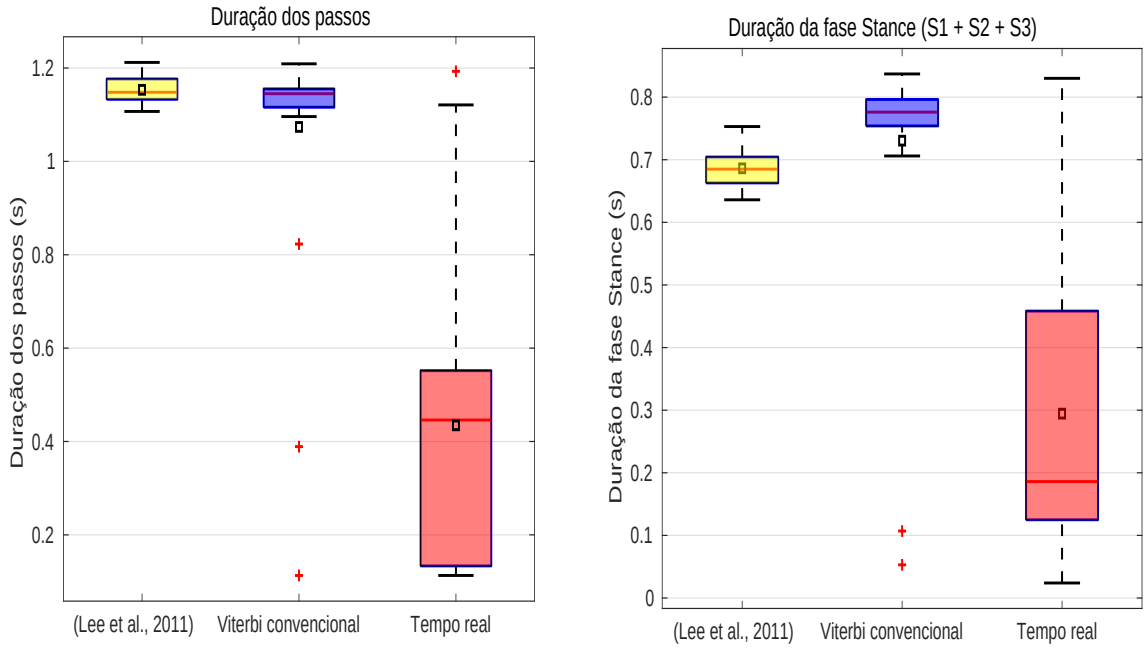
Fonte: O Autor (2020)

Figura 16: Comparação entre os estados estimados em tempo real e em pós-processamento. Destacam-se a taxa de inserções elevada e o atraso dos estados estimados em tempo real.



Fonte: O Autor (2020)

Figura 17: *Boxplot* da duração dos passos e da fase *Stance* para cada um dos algoritmos



(a) Duração dos passos

(b) Duração da fase *Stance*

Fonte: O Autor (2020)

Nota: A fase *Stance* corresponde a todo o tempo em que o pé está em contato com o solo. A cor das caixas corresponde ao algoritmo: Lee *et al.*, 2011 (amarelo), algoritmo de Viterbi em pós-processamento com dados filtrados (azul) e algoritmo de Viterbi em tempo real com janela deslizante de tamanho fixo (vermelho). Os traços vermelhos representam as medianas. Os quadrados pretos são as médias. As caixas são limitadas pelos quartis inferior e superior, o que representa 50% dos valores observados. Os tracejados correspondem ao intervalo que conteria aproximadamente 99.3% dos dados caso a distribuição fosse normal. As cruzes vermelhas correspondem aos *outliers*.

4.2.1 Performance e comparação com outros trabalhos da literatura

Apenas o modelo oculto de Markov são introduzidas as métricas de desempenho P (Precisão), R (*Recall*) e F_1 (F_1-*score*), sendo que F_1 leva em conta tanto a Precisão quanto o *Recall*:

$$P = \frac{TP}{TP + FP} \quad , \quad R = \frac{TP}{TP + FN} \quad , \quad F_1 = \frac{2 \cdot P \cdot R}{P + R} \tag{4.1}$$

em que TP é o número de verdadeiros positivos, FN o número de falsos negativos e FP o número de falsos positivos. Temos P, R e $F_1 \in (0, 1)$. Quanto mais próximos de 1 melhor a performance. Foram considerados verdadeiros positivos os passos completos cujos eventos HS (*Heel-Strike*) e TO (*Toe-Off*) estavam dentro de uma janela de tolerância de 200 ms em relação aos eventos IC (*Initial-Contact*) e EC (*End-Contact*), respectivamente, detectados com o algoritmo de (LEE; PARK, 2011). Além disso foi realizada uma inspeção visual para confirmar que o algoritmo de (LEE; PARK, 2011) detectou todos os passos corretamente. Todos os demais passos completos inseridos dentro dos passos verdadeiros positivos foram considerados falsos positivos. Não houve falsos negativos, que seriam os casos em que um passo é completamente ignorado pelo algoritmo.

Os índices obtidos com os métodos utilizados neste trabalho são mostrados na Tabela 1.

Tabela 1: Precisão, *Recall* e F_1-*score* obtidos para cada um dos algoritmos

Método	P	R	F_1
(Lee et al., 2011)	1.00	1.00	1.00
Pós-processamento - Dados não filtrados	0.71	1.00	0.83
Pós-processamento - Dados filtrados	0.93	1.00	0.96
Tempo real	0.38	1.00	0.55

Fonte: O Autor (2020)

No caso do pós-processamento com dados filtrados, a acurácia foi similar a outros métodos disponíveis na literatura, que reportam acurácia próxima a 100% e $F_1 > 0.9$. Já em relação aos estados estimados em tempo real, o desempenho foi inferior a outros métodos da literatura. Por exemplo (Pérez-Ibarra; Siqueira; Krebs, 2020) obtiveram um índice F_1 de 0.99 para indivíduos sem debilidade na marcha, enquanto que aqui foi de 0.55. Já (MANNINI; GENOVESE; SABATINI, 2013) obtiveram taxas de inserção menores que 0.50% para todos os quatro eventos, enquanto que aqui a taxa de inserção foi muito elevada (superior a 100%). Por outro lado, a taxa de eliminação (quando um ciclo de marcha é ignorado) observada aqui foi igual à de (MANNINI; GENOVESE; SABATINI, 2013), ou seja 0%, pois nenhum ciclo de marcha foi ignorado. Isso corresponde a $R = 1$, pois não há falsos negativos.

As medianas da duração dos passos e das fases *Stance* obtidas com os algoritmos são mostradas na Tabela 2. Novamente a performance do algoritmo de Viterbi convencional com os dados filtrados foi muito similar ao algoritmo de Lee *et al.*. Isso também fica claro na Figura 17, em que a distribuição das durações ficou próxima ao algoritmo de Lee *et al.*.

Observa-se que as inserções configuraram *outliers* visto que elas são bastante discrepantes em relação aos outros passos. Por outro lado o algoritmo em tempo real - Viterbi em uma janela deslizante de tamanho fixo - foi bastante afetado pelas inserções, que levaram a mediana das durações a valores reduzidos, visto que as durações das inserções são curtas.

Além disso, dada a alta taxa de inserções, elas não configuraram *outliers* pois elas foram a maioria dos passos detectados. Isso leva à conclusão de que é necessário adicionar uma estratégia que impeça a detecção de ciclos de marcha com duração muito curta. Essa estratégia foi adotada em (MANNINI; SABATINI, 2011), em que ciclos completos com duração menor que 0.35 s eram descartados.

Tabela 2: Medianas da duração dos passos e da fase *Stance* obtidas para cada um dos algoritmos

Método	Duração dos passos [s] (Mediana)	Duração das fases *Stance* [s] (Mediana)
(Lee et al., 2011)	1.15	0.69
Pós-processamento - Dados filtrados	1.15	0.78
Tempo real	0.45	0.19

Fonte: O Autor (2020)

5 DISCUSSÃO GERAL E CONCLUSÕES

Este trabalho está inserido num contexto em que a detecção acurada de fases do caminhar é relevante para tratamentos de reabilitação motora como estimulação elétrica funcional, controle de sistemas robóticos de suporte à marcha ou reabilitação. Nesses casos são necessários algoritmos em tempo real com baixo atraso. Por outro lado, o acompanhamento a longo prazo de pacientes em tratamento de reabilitação motora é um exemplo de aplicação que permite o pós-processamento. Algumas métricas como duração das fases da marcha podem dar indícios sobre os graus de debilidade e de recuperação em um tratamento. Em todos as aplicações é muito vantajosa a utilização de *smartphones*, pois grande parte da população conta com tal aparelho. Dessa forma, este trabalho sugere a disponibilização desses algoritmos em aplicativos de celular.

Uma desvantagem do método utilizado aqui no modelo das quatro fases foi a necessidade de acoplamento do aparelho celular sobre o peito do pé, o que dificultaria uma aplicação real. Nesse sentido, faz-se necessário investigar o posicionamento do celular em outros locais, como a coxa, até mesmo dentro de um bolso, sem que ele esteja afixado junto ao corpo. Um método acurado de detecção do caminhar humano em tempo real utilizando um aparelho celular no bolso representaria um grande avanço nesta área.

Pode-se considerar que os objetivos principais deste projeto foram atingidos com êxito. Foram derivadas duas matrizes de transição do caminhar humano, uma para cada modelo.

Em relação ao modelo das articulações divididas em setores, derivou-se uma MPT que foi útil para o projeto de um controlador Markoviano para um exoesqueleto. Esses estados poderiam ser facilmente detectados em tempo real, uma vez que bastaria simplesmente identificar em qual setor cada articulação está. A combinação entre os setores da articulação corresponde diretamente ao estado. Este método, no entanto, não leva em conta uma sequência de observações para determinar o estado em determinado instante de tempo, o que pode produzir sequências improváveis caso haja ruído nos dados.

Para o modelo das quatro fases da marcha com sentido físico, foi possível estimar os estados do caminhar humano tanto em pós-processamento quanto em tempo real. Ademais, os estados com sentido físico que foram escolhidos com base na literatura são de simples identificação através um conjunto de regras para a fase de treinamento, o que possibilitou o avanço do trabalho mesmo sem acesso aos equipamentos do laboratório no período de isolamento social devido à pandemia de SARS-CoV-2. Isso sugere que é possível treinar modelos relativamente satisfatórios com equipamentos disponíveis para grande parte da população e em ambientes diversos. Faz-se necessário, contudo, melhor investigar essa

hipótese.

A seguir são apresentadas algumas limitações dos métodos empregados bem como dos resultados obtidos. Nos experimentos não foram utilizados dados de indivíduos com marcha debilitada, cujos padrões de caminhada podem variar significativamente em relação ao padrão da marcha em indivíduos sem debilidade no caminhar. Provavelmente os resultados seriam degradados. Ademais o conjunto de dados foi pequeno nos experimentos relativos ao modelo das quatro fases, o que implica numa baixa significância estatística.

Além disso, vale ressaltar que o processamento em tempo real se deu em um *laptop* com um processador Intel Core i5 de oitava geração. Caso o processamento se desse no próprio aparelho celular, possivelmente os algoritmos consumiriam grande parte dos recursos do dispositivo, ou seriam até inviáveis dependendo do aparelho. Nesse sentido, a elaboração de algoritmos poucos custosos é essencial.

Uma vantagem da abordagem aqui utilizada é que o atraso dos estados estimados em tempo real pelo algoritmo de Viterbi em uma janela limitada de tamanho fixo é aproximadamente constante em relação aos estados estimados em pós-processamento pelo algoritmo de Viterbi tradicional, por conta do tamanho fixo do janelamento. Entretanto, não existe garantia de otimalidade em relação ao critério da Equação 2.8. Já o Algoritmo de Viterbi *On-line* garantiria tal condição, porém com a desvantagem de o atraso ser incerto, variável e possivelmente ilimitado. Para contornar isso seria também necessário limitar o tamanho dos caminhos de estados ocultos, o que implicaria na perda da garantia de otimalidade da Equação 2.8.

5.1 Trabalhos futuros

Em primeiro lugar os resultados em pós-processamento levantam a pergunta natural de qual é a quantidade de dados mínima necessária para realizar o treinamento e produzir um modelo satisfatório. Além disso encontrar formas de facilitar o posicionamento dos sensores é uma área importante a ser estudada.

Algumas ideias para melhoria dos resultados e investigação de outras técnicas decorrem diretamente da literatura referente à modelagem do caminhar humano por modelos de Markov.

Primeiramente, deve-se ressaltar que é recomendada a utilização de métodos mais precisos na fase de etiquetamento dos estados de treinamento, como os que são considerados padrão ouro: sensores de pressão na sola do calçado e sistemas optoeletrônicos. Isso permitiria uma melhor validação quantitativa do modelo. Outro possível aprimoramento dos resultados poderia ser obtido com a aplicação do Algoritmo de Viterbi *On-line* para a estimação dos estados em tempo real, ao invés do método utilizado que consistiu em aplicar o Algoritmo de Viterbi convencional em uma janela deslizante de tamanho fixo.

Além disso, modelos mais complexos do caminhar humano podem ser testados, como por exemplo os modelos multivariados, que são um caso mais geral do que o modelo gaussiano adotado neste trabalho. Não obstante, foi adotada uma única saída como vetor de observação. É de se esperar que um vetor contendo mais saídas, combinando dados de mais IMUs ou IMUs combinadas com sensores de pressão produza uma caracterização mais completa do caminhar humano.

REFERÊNCIAS

BISHOP, C. M. **Pattern Recognition and Machine Learning (Information Science and Statistics)**. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN 0387310738.

CATALFAMO, P.; GHOUSSAYNI, S.; EWINS, D. Gait event detection on level ground and incline walking using a rate gyroscope. **Sensors**, v. 10(6):5683-5702, 2010.

INOUE, R. S. **Controle Robusto Descentralizado de Movimentos Coordenados de Robôs Heterogêneos**. 2012. Tese (Doutorado) — Escola de Engenharia de São Carlos, Universidade de São Paulo, 2012.

JILKOV, V.; LI, X. Online Bayesian estimation of transition probabilities for Markovian jump systems. **IEEE Transactions on Signal Processing**, v. 52, n. 6, p. 1620–1630, June 2004. ISSN 1053-587X.

LEE, J. K.; PARK, E. J. Quasi real-time gait event detection using shank-attached gyroscopes. **Medical Biological Engineering Computing volume**, v. 49, p. 707 – 712, 2011.

MANNINI, A.; GENOVESE, V.; SABATINI, A. Online decoding of hidden markov models for gait event detection using foot-mounted gyroscopes. **Biomedical and health informatics, IEEE journal of**, PP, p. (online ahead of print), 12 2013.

MANNINI, A.; SABATINI, A. M. A hidden markov model-based technique for gait segmentation using a foot-mounted gyroscope. **Annual International Conference of the IEEE EMBS**, Boston, Massachusetts USA, v. 33, p. 4369–4373, 2011.

MARIANI, B. et al. Quantitative estimation of foot-flat and stance phase of gait using foot-worn inertial sensors. **Gait Posture**, v. 37, n. 2, p. 229 – 234, 2013. ISSN 0966-6362. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0966636212002822>.

MITSCHKA, C. M. et al. Recursive linear quadratic regulator subject to markovian jump linear systems in a robotic application system for rehabilitation. **XXI Congresso Brasileiro de Automática - CBA2016**, p. 840–844, Outubro 2016. Disponível em: <http://www.swge.inf.br/proceedings/paper/?P=CBA2016-0254>.

MURPHY, K. [S.l.], 1998. Disponível em: <https://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html>.

NIELSEN, S. F. **Continuous-time homogeneous Markov chains**: Stochastic processes. [S.l.]: University of Copenhagen, Department of Mathematical Sciences, 2009. Disponível em <http://web.math.ku.dk/~susanne/kursusstokproc/ContinuousTime.pdf>. Acesso em: 20 dez. 2019.

ORGUNER, U.; DEMIREKLER, M. Maximum Likelihood Estimation of Transition Probabilities of Jump Markov Linear Systems. **IEEE Transactions on Signal Processing**, v. 56, n. 10, p. 5093–5108, Oct 2008. ISSN 1053-587X.

PARDOUX Étienne. **Processus de Markov et applications**: Algorithmes, réseaux, génome et finance. 2006. Disponível em <http://paestel.fr/sites/default/files/Mat_les_ressources/M1/Cours/E.%20Pardoux_493.pdf>. Acesso em: 20 dez. 2019.

PFEAU, T.; WELLE, R. Comparison of a standalone consumer grade smartphone with a specialist inertial measurement unit for quantication of movement symmetry in the trotting horse. **Equine Veterinary Journal**, v. 49(1):124-129, 2015. ISSN 0425-1644.

Pérez-Ibarra, J. C.; Siqueira, A. A. G.; Krebs, H. I. Identification of gait events in healthy and parkinson's disease subjects using inertial sensors: A supervised learning approach. **IEEE Sensors Journal**, p. 1–1, 2020.

Pérez-Ibarra, J. C. et al. Real-time identification of impaired gait phases using a single foot-mounted inertial sensor: Review and feasibility study. In: **2018 7th IEEE International Conference on Biomedical Robotics and Biomechatronics (Biorob)**. [S.l.: s.n.], 2018. p. 1157–1162.

Rabiner, L. R. A tutorial on hidden markov models and selected applications in speech recognition. **Proceedings of the IEEE**, v. 77, n. 2, p. 257–286, 1989.

RUETERBORIES, J. et al. Methods for gait event detection and analysis in ambulatory systems. **Medical Engineering Physics**, v. 32, p. 545–552, 2010.

SABATINI, A. M. Kalman-filter-based orientation determination using inertial/magnetic sensors: Observability analysis and performance evaluation. **Sensors (Basel, Switzerland)**, v. 11, p. 9182–206, 12 2011.

SILVA, A. B. N. **Um modelo de unidade de medida inercial utilizando três acelerômetros triaxiais**. Agosto 2013. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Norte, Agosto 2013.

VU, H. T. T. et al. A review of gait phase detection algorithms for lower limb prostheses. **Sensors**, v. 20(14):3972, 2020.

WANG, G. ML Estimation of Transition Probabilities in Jump Markov Systems via Convex Optimization. **IEEE Transactions on Aerospace and Electronic Systems**, v. 46, n. 3, p. 1492–1502, July 2010. ISSN 0018-9251.

XSENS Technologies B.V., *MTw User Manual*. Enschede, The Netherlands, 2013.

Zhi, X.; Xu, Q.; Schwertfeger, S. Evaluation of Smartphone IMUs for Small Mobile Search and Rescue Robots. **arXiv e-prints**, p. arXiv:1912.01221, dez. 2019.

ŠRáMEK, R. **The on-line Viterbi algorithm**. 2007. Dissertação (Mestrado) — Department of Computer Science Faculty of Mathematics, Physics and Informatics Comenius University Bratislava, 2007. Disponível em: <http://www.compbio.fmph.uniba.sk/papers/07sramekth.pdf>.

ŠRáMEK, R.; BREJOVá, B.; VINAř, T. **On-Line Viterbi Algorithm for Analysis of Long Biological Sequences**. Berlin, Heidelberg: Giancarlo R., Hannenhalli S. (eds) Algorithms in Bioinformatics. WABI 2007. Lecture Notes in Computer Science, Springer-Verlag, 2007. v. 4645. ISBN 978-3-540-74125-1.

Apêndices

APÊNDICE A – CÓDIGOS EM MATLAB

Códigos para a simulação de uma cadeia de Markov simples.

```
%Cadeia de Markov – Exemplo de Sistema simples
states = [];
states(1) = 1;
for i = 2 :21
    if states(i-1) == 1
        states(i) = 1
        if rand > 0.7
            states(i) = 2;
        else
        end
    else
         states(i) = 1;
    end
    i = i + 1;
end
plot((0:1:20),states, 'x', 'LineWidth',30)
axis([0 20 0.5 2.5])
yticks([1 2]);
xticks([0:1:20]);
ylabel('estados')
xlabel('instantes de tempo')

n1 = 0; n2 = 0;
P = zeros(2,2);
n1 = 1;

%Conta o numero de vezes que esteve em cada estado
for i = 2 :21
    if  states(i) == 1
        n1 = n1 + 1;
        if  states(i-1) == 1
            P(1,1) =  P(1,1) + 1;
        else
            P(2,1) =  P(2,1) + 1;
        end
```

```
    else
        n2 = n2 + 1;
        if  states(i-1) == 1
            P(1,2) =  P(1,2) + 1;
        else
            P(2,2) =  P(2,2) + 1;
        end
    end
end
if states(end) == 1
    n1 = n1 - 1;
else
    n2 = n2 - 1;
end
P(1,:) = P(1,:)/n1;
P(2,:) = P(2,:)/n2;
```

Código para treinamento do Modelo Oculto de Markov.

```
clear all
close all

load('plano_05_05_2020_pe_direito.mat')
data = plano_05_05_2020_pe_direito;
load('plano_05_05_2020_pe_direito_data3.mat')

data2 = data1588619683(:,5)*360/(2*pi);
t2 = data1588619683(:, 1);
t2 = t2- t2(1);
t2 =t2/1000 ;
t = data(:, 1);
t = t- t(1);
t =t/1000 ;
n = length(t);
avgT = t(n)/(n-1);
avgF = 1/avgT %Average frequency in Hz
angular_velocity =  data(:,5)*360/(2*pi);
[b,a] = butter(2,15/(avgF/2));
y = filter(b,a,angular_velocity);
data2filtered = filter(b,a,data2);

```

```
23 angular_velocity_filtered =y;
24 difference = abs(angular_velocity - angular_velocity_filtered);
25 zci = @(v) find(v(:).*circshift(v(:), [-1 0]) <= 0);
26 % Returns Zero-Crossing Indices Of Argument Vector
27 zx = zci(y);
28 %acc = abs([diff(angular_velocity); 0])
29 %angular_position = cumtrapz(t, angular_velocity);
30
31 angular_position_filtered = cumtrapz(t, angular_velocity_filtered);
32 angular_acceleration = [diff([0; angular_velocity_filtered])/avgT];
33 lambdaFF = -50; %FF threshold in   /s
34 lambdaHO = -50; %HO threshold in   /s
35 %To find zeros
36 %angular_acceleration this > 2000; angular_accelerationnext <-1400
37 HS = find(difference > 5);
38
39 FFindicesboolean = zeros(1,n);
40 HOindicesboolean = zeros(1,n);
41 TOindicesboolean = zeros(1,n);
42 HSindicesboolean = zeros(1,n);
43 j = 1;
44 for i=1 :length(zx)
45     tzeros(i) = t(zx(i));
46     yzeros(i) = y(zx(i));
47     %tzeros2(i) = t(zx(i)) + avgT;
48     %yzeros2(i) = y(zx(i) + 1);
49     if i < length(zx)
50         if (angular_acceleration(zx(i)) > 2000 &&
51         angular_acceleration(zx(i +1)) <-1400)
52             tzerosTO(j) = t(zx(i));
53             yzerosTO(j) = y(zx(i));
54             indicesTO(j) = zx(i);
55             TOindicesboolean(1,zx(i)) = 1;
56             tzerosnext(j) = t(zx(i+1));
57             yzerosnext(j) = y(zx(i+1));
58             indicesnext(j) = zx(i+1);
59             j = j +1;
60         end
61     end
62 end
63
64 j = 1;
```

```
k = 1;
for i=2:length(angular_velocity_filtered)
    if (angular_velocity_filtered(i) >= lambdaFF &&
    angular_velocity_filtered(i-1) < lambdaFF &&
    angular_position_filtered(i)> -15)
        FFindices(j) = i;
        FFindicesboolean(1,i) = 1;
        yFF(j) = angular_velocity_filtered(i);
        tFF(j) = t(i);
        j = j +1;
    end
      if (angular_velocity_filtered(i) <= lambdaFF &&
      angular_velocity_filtered(i-1) > lambdaFF &&
      angular_position_filtered(i) < 0 )
        HOindices(k) = i;
        HOindicesboolean(1,i) = 1;
        yHO(j) = angular_velocity_filtered(i);
        tHO(j) = t(i);
        k = k+1;
      end
end
HSindicesboolean = zeros(1,n);

FFindices = FFindices(1,5:end);
yFF = yFF(1,5:end);
tFF = tFF(1,5:end);
for i=1:length(FFindices)
    [m, HSindices(i)] = max(difference(indicesnext(i):FFindices(i)));
    HSindices(i) = HSindices(i) + indicesnext(i) - 1;
    yHS(i) = angular_velocity_filtered(HSindices(i));
    tHS(i) = t(HSindices(i));
    HSindicesboolean(HSindices(i)) = 1;
end
states = [];
state = 1;
for i=1 :length(angular_velocity_filtered)
    if HSindicesboolean(i)
        state = 1;
    elseif FFindicesboolean(i)
        state = 2;
    elseif HOindicesboolean(i)
        state = 3;
```

```
107     elseif TOindicesboolean(i)
108         state = 4;
109     end
110     states(i) = state;
111 end
112 
113 % scatter(tzerosTO, yzerosTO);
114 % scatter(tzerosnext, yzerosnext);
115 % scatter(tFF, yFF);
116 % scatter(tHS, yHS);
117 % % scatter(tHO, yHO);
118 % % %scatter(tzeros2, yzeros2)
119 %  plot(t, angular_velocity)
120 % % %plot(t, angular_velocity)
121 % % plot(t, difference, 'r')
122 % % plot(t, angular_acceleration, 'g')
123 
124 % % %plot(t, acc)
125 
126 % plot(t,angular_position_filtered)
127 % plot(t,100*states)
128 
129 %plot(states)
130 
131 training_states = states(1501:2589);
132 training_cycles = 6;
133 testing_cycles = 3;
134 testing_states = states(2590:3155);
135 
136 training_emissions = angular_velocity(1501:2589);
137 testing_emissions =  angular_velocity(2590:3155);
138 
139 emissions_1 = [];
140 emissions_2 = [];
141 emissions_3 = [];
142 emissions_4 = [];
143 
144 for i=1 :length(training_states)
145     if training_states(i) == 1
146         emissions_1 =  [emissions_1 training_emissions(i)];
147     elseif training_states(i) == 2
148         emissions_2 =  [emissions_2 training_emissions(i)];
```

```
149     elseif training_states(i) == 3
150         emissions_3 =  [emissions_3 training_emissions(i)];
151     elseif training_states(i) == 4
152         emissions_4=  [emissions_4 training_emissions(i)];
153     end
154 
155 end
156 emissions_stds= [std(emissions_1);
157     std(emissions_2);
158     std(emissions_3);
159     std(emissions_4)];
160 
161 emissions_avgs = [mean(emissions_1);
162     mean(emissions_2);
163     mean(emissions_3);
164     mean(emissions_4)];
165 a12 = training_cycles/length(emissions_1);
166 a23 = training_cycles/length(emissions_2);
167 a34 = training_cycles/length(emissions_3);
168 a41 = training_cycles/length(emissions_4);
169 A = [(1-a12) a12 0 0;
170     0 (1-a23) a23 0;
171     0 0 (1-a34) a34;
172     a41 0 0 (1-a41)];
173 addpath(genpath('/home/francisco/Documents/MATLAB/TCC/HMMlibrary/HMM/HMMall'))
174 Sigma = zeros(1,1,4);
175 for i=1:4
176     Sigma(1,1,i) = emissions_stds(i);
177 end
178 B = mixgauss_prob(testing_emissions',emissions_avgs', Sigma);
179 prior = [0.25; 0.25; 0.25; 0.25];
180 [path] = viterbi_path(prior, A, B);
181 totalstates = [zeros(1,1500), training_states, path];
182  title('Estados de Markov ')
183 xlabel('Tempo (s)')
184 hold
185 figure(1)
186 hold on
187  plot(angular_velocity_filtered)
188  plot(100*totalstates, 'r', 'linewidth',2, 'DisplayName',
189  'Estados estimados com modelo de Markov')
190  plot(100*states, 'g','linewidth',4,'DisplayName',
```

```
191  'Estados estimados com modelo de Markov')
192 %  scatter(tFF, yFF);
193 % scatter(tHS, yHS);
194 % scatter(tHO, yHO);
195 %     legend show
196 % hold off
197 B2 = mixgauss_prob(data2',emissions_avgs', Sigma);
198 [path2] = viterbi_path(prior, A, B2);
199 figure(2)
200 B2filtered = mixgauss_prob(data2filtered',emissions_avgs', Sigma);
201 [path2filtered] = viterbi_path(prior, A,B2filtered);
202 plot(t2, 100*path2)
203 plot(t2, data2filtered)
```

APÊNDICE B – CÓDIGO EM PYTHON

Código utilizado para a recepção, gravação e processamento dos dados do celular em tempo real.

```
import socket, traceback
import string
import time
import datetime as dt
import keyboard
import matplotlib.pyplot as plt

import datetime as dt
import matplotlib.animation as animation
import threading
import os
import numpy as np

from hmmlearn import hmm
from scipy import signal

data = (0, 0, 0, 0, 0 ,0 ,0)
delay = 0
loop_flag = True
def stop_loop():
    global loop_flag
    keyboard.wait(" ")
    loop_flag = False
    print("endloop")

def get_delay():
    global  data
    while True:  # making a loop
        try:  # used try so that if user pressed other than the given key er
            if keyboard.is_pressed(' '):  # if key 'space' is pressed
                #print('You Pressed A Key!')
                delay = time.time()
                print(delay)
                while data[5] > -0.04:
```

```
35                      pass
36                  delay = time.time() - delay
37                  print(delay)
38                  break  # finishing the loop
39          except:
40              break
41      delay = time.time()
42
43
44  def sensor_function():
45
46  fs = 165.9264858735708
47  fc = 15  # Cut-off frequency of the filter
48  w = fc / (fs / 2)  # Normalize the frequency
49  b, a = signal.butter(2, w, 'low')
50
51      model = hmm.GaussianHMM(n_components=4,
52                              covariance_type="spherical", init_params="cm",
53                              params="cmt",
54                              algorithm="viterbi")
55
56      model.startprob_ = np.array([0.25, 0.25, 0.25, 0.25])
57      model.transmat_ = np.array([[0.948275862068966, 0.0517241379310345, 0,  0],
58                              [0, 0.984732824427481,  0.0152671755725191, 0],
59                              [0, 0,  0.969543147208122,  0.0304568527918782],
60                              [0.0156657963446475,    0,  0,  0.984334203655353]])
61
62      model.means_ = np.array([[-199.731014627276],
63                               [-18.5162706325788],
64                               [-298.191241225539],
65                               [232.572872839484]])
66
67      model.covars_ = np.array([113.830919434039,
68                                14.8864353839467,
69                                221.608481988583,
70                                163.783635242862])
71
72      global loop_flag
73      host = ''
74      port = 50000
75
76      s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
```

```
77      s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
78      s.setsockopt(socket.SOL_SOCKET, socket.SO_BROADCAST, 1)
79      s.bind((host, port))
80
81      # used for debugging
82      saved_data = []
83      angular_velocity = []
84      print("Success binding")
85      global data
86      last_output_state = 0
87
88      try:
89
90          # Set up plot to call animate() function periodically
91          while loop_flag:
92              #print(loop_flag)
93              #start = time.time()
94              message, address = s.recvfrom(8192)
95              m = message.decode("utf-8")
96              #elapsed_time_fl = (time.time() - start)
97              #freq = 1 / elapsed_time_fl
98
99              data = (m.split('<TimeStamp>')[1].split('</TimeStamp')[0],
100               m.split('<Accelerometer1>')[1].split('</Accelerometer1>')[0],
101               m.split('<Accelerometer2>')[1].split('</Accelerometer2>')[0],
102               m.split('<Accelerometer3>')[1].split('</Accelerometer3>')[0],
103               m.split('<Gyroscope1>')[1].split('</Gyroscope1>')[0],
104               m.split('<Gyroscope2>')[1].split('</Gyroscope2>')[0],
105               m.split('<Gyroscope3>')[1].split('</Gyroscope3>')[0]
106                 )
107             # print(data)
108             angular_velocity.append([float(data[4])*360/(2*np.pi)])
109             saved_data.append(data)
110             start_prob = np.array([0.0, 0.0, 0.0, 0.0])
111             start_prob[last_output_state] = 1.0
112             model.startprob_ = start_prob
113             if len(angular_velocity) >= 100:
114                 output_signal = signal.filtfilt(b, a, angular_velocity[-15:
115             else:
116                 output_signal = angular_velocity[-15:]
117             [p, output_data] = model.decode(output_signal)
118             if len(output_data) >=2:
```

```
119                 last_output_state = output_data[1]
120             print(last_output_state)
121
122         print("Finished acquisition")
123         print("Saving data")
124
125         file_path = '<YOUR_FILE_PATH>'
126         with open(file_path, 'w') as fp:
127             fp.write('\n'.join('{} {} {} {} {} {} {}'.format(x[0],
128             x[1], x[2], x[3], x[4], x[5], x[6]) for x in saved_data))
129             print("Finished saving sata")
130     except (KeyboardInterrupt, SystemExit):
131         print("end")
132         pass
133
134
135
136
137
138 # This function is called periodically from FuncAnimation
139 def animate(i, xs, ys1, ys2, ys3):
140     aux1 = round(float(data[4]), 2)
141     aux2 = round(float(data[5]), 2)
142     aux3 = round(float(data[6]), 2)
143
144     # Add x and y to lists
145     xs.append(dt.datetime.now().strftime('%H:%M:%S.%f'))
146     ys1.append(aux1)
147     ys2.append(aux2)
148     ys3.append(aux3)
149     if keyboard.is_pressed('p'):
150         while True:
151             pass
152
153     # Limit x and y lists to 20 items
154     xs = xs[-20:]
155     ys1 = ys1[-20:]
156     ys2 = ys2[-20:]
157     ys3 = ys3[-20:]
158
159     # Draw x and y lists
160     ax.clear()
```

```
161     ax.plot(xs, ys1, 'r')  #Lateral (Direita +)
162     ax.plot(xs, ys2, 'b')  #Frente-tr s (Frente +)
163     ax.plot(xs, ys3, 'g')  #Vertical (Cima +)
164
165
166
167     # Format plot
168     plt.xticks(rotation=45, ha='right')
169     plt.subplots_adjust(bottom=0.30)
170     plt.title('Angular velocity over Time')
171     plt.ylabel('Angular velocity (rad/s)')
172
173
174
175 if __name__ == "__main__":
176
177     x = threading.Thread(target=sensor_function, args=())
178     x.daemon = True
179     x.start()
180     end_loop_thread = threading.Thread(target=stop_loop, args=())
181     end_loop_thread.daemon = True
182     end_loop_thread.start()
183
184     fig = plt.figure()
185     ax = fig.add_subplot(1, 1, 1)
186     # Create figure for plotting
187
188     xs = []
189     ys1 = []
190     ys2 = []
191     ys3 = []
192     print("Creating animation")
193     ani = animation.FuncAnimation(fig, animate, fargs=(xs,ys1, ys2, ys3), in
194     plt.show()
```