Pós-Graduação em Ciência da Computação

Marcus Felipe Raele Rios

Estratégia de Localização para Sistemas de Acesso Passivo a Veículos Utilizando Sensores Inercias de *Smartphones*

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
http://cin.ufpe.br/~posgraduacao

Recife
2020

Marcus Felipe Raele Rios

Estratégia de Localização para Sistemas de Acesso Passivo a Veículos Utilizando Sensores Inercias de *Smartphones*

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Inteligência Computacional
Orientador: Prof. Abel Guilhermino

Recife
2020

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

R586e Rios, Marcus Felipe Raele
Estratégias de localização para sistemas de acesso passivo a veículos utilizando sensores inerciais de *smartphones* / Marcus Felipe Raele Rios. – 2020.
106 f.: il., fig., tab.

Orientador: Abel Guilhermino da Silva Filho.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2020.
Inclui referências.

1. Inteligência computacional. 2. Aprendizagem de máquina. I. Silva Filho, Abel Guilhermino (orientador). II. Título.

006.31 CDD (23. ed.) UFPE - CCEN 2020 - 118

Marcus Felipe Raele Rios

“Estratégia de Localização para Sistemas de Acesso Passivo a Veículos Utilizando Sensores Inercias de Smartphones”

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Aprovado em: 12 de fevereiro de 2020.

BANCA EXAMINADORA

_____

Prof. Dr. Daniel Carvalho da Cunha
Centro de Informática / UFPE

_____

Prof. Dr. Wellington Pinheiro dos Santos
Engenharia Biomédica/UFPE

_____

Prof. Dr. Abel Guilhermino da Silva Filho
Centro de Informática / UFPE
(**Orientador**)

Dedico este trabalho a mim mesmo e a todos que acreditam em mim.

AGRADECIMENTOS

Agradeço aos meus amigos do LIVE pelos debates acerca deste trabalho. Aos meus colegas de trabalho do ISI-TICs, que mesmo eu passando pouco tempo no ISI, me deram dicas sobre a escrita do trabalho. E aos meus amigos da época da graduação no CIn-UFPE, pelos momentos de apoio e descontração que me deram.

RESUMO

Com os avanços tecnológicos na indústria automotiva os meios de acesso ao veículo foram se modernizando. Inicialmente a forma de se entrar no veículo era através de uma chave mecânica não autenticada, que era possível se fazer ligação direta para ligar o veículo. Com o passar do tempo surgiram os *immobilisers* que serviam pra autenticar a chave, reduzindo o número de roubos de veículos. Mais a frente surgiram os sistemas *Remote Keyless Entry* (RKE) e *Passive Keyless Entry and Start* (PKES), o primeiro permite a abertura do veículo a distância através de ondas de rádio e o segundo a abertura de forma passiva do veículo (sem a necessidade de se apertar nenhum tipo de botão, apenas estar portando a chave) também utilizando ondas de rádio. Essas novas tecnologias além de trazer mais conforto para os usuários também trouxeram mais desafios, dentre eles o ataque de *relay*, que consiste em um ataque de camada física muito difícil de ser detectado e de se proteger dele. Com o advento dos *smartphones*, uma nova possibilidade surgiu, utilizar os *smartphones* como chave passiva de veículos. Este tipo de tecnologia já encontrada em carros de luxo como por exemplo Tesla Model 3, que é possível utilizar apenas o *smartphone* como chave passiva do carro. E não diferente de sistemas PKES de chaves clássicas de veículos os *smartphones* são vulneráveis a ataques de *relay*. Diante desta situação, este trabalho propõe um algoritmo de localização 2D baseado em *Pedestrian Dead Reckoning* (PDR) que pode ser utilizado em um sistema de chave passiva para *smartphones*. Com este algoritmo proposto o sistema seria capaz de controlar os momentos em que o rádio da chave estaria ligado impedindo os ataques de *relay*, uma vez que, apenas quando o usuário estivesse realmente perto do veículo o serviço de abertura do veículo seria habilitado. Neste sistema de PDR são empregadas técnicas de aprendizagem de máquina para fazer reconhecimento de posições e orientações do *smartphone*, assim como, para fazer estimação de deslocamento em cada passo dado pelo usuário. Foi verificado que técnicas baseadas em árvores possuem uma boa acurácia para fazer estimação dos passos e SVM são boas para definir as posições e orientações do *smartphone*. O sistema proposto se mostrou eficiente, erros inferiores a $2m$, em sua maioria para trajetos curtos. Para trajetos longos o sistema não apresentou um desempenho satisfatório, devido a um erro de *drift* não considerado pelo algoritmo de direção proposto.

Palavras-chaves: *Passive Keyless Entry and Start*. *Pedestrian Dead Reckoning*. Localização *Indoor*. Localização *Outdoor*. Sensores. Aprendizagem de Máquina.

ABSTRACT

Due to technological advances in the automotive industry, the ways to access a vehicle also evolved. Initially, one way to get into the vehicle was through an unauthenticated mechanical key, which was possible to do hotwire to start the vehicle. Over time emerged the immobilisers that served to authenticate the key, reducing the number of vehicle thefts. Later came the *Remote Keyless Entry* (RKE) and *Passive Keyless Entry and Start* (PKES) systems, the first allowing the vehicle to be opened at a distance through radio waves and the second to passively opening the vehicle (without the need to push any kind of button, just bearing the key) also using radio waves. These new technologies not only bring more comfort to users but also bring more challenges, between it the relay attack, which is a physical layer attack that is very difficult to detect and protect against. Due to technological improvements in smartphones, the new possibility has emerged, using smartphones as a passive vehicle key. This type of technology is already found in luxury cars, such as Tesla Model 3, which is possible to use a smartphone as a passive car key. And no different from classical PKES keys, smartphones are vulnerable to relay attacks. Given this situation, this work proposes a 2D localization algorithm based on *Pedestrian Dead Reckoning* (PDR) that can be used in a passive smartphone key system. Using the proposed algorithm the system would be able to control when the smartphone radio would be on, preventing relay attacks, since only when the user is near to the vehicle, the opening service of the vehicle is enabled. In the PDR system prosposed, machine learning techniques are employed to make position and orientation recognition of the smartphone, as well to estimate displacement at each step given by the user. Tree-based techniques have been found to have good accuracy for estimating steps and SVM for defining smartphone positions and orientations. The proposed system proved to be efficient, errors less than 2 meters, mostly for short distances. For long journeys the system did not perform well due to a drift error not considered by the proposed heading algorithm.

Keywords: *Passive Keyless Entry and Start*. *Pedestrian Dead Reckoning*. Indoor Localization. Outdoor Localization. Sensors. Machine Learning.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

ANN	*Artificial Neural Network*
API	*Application Programming Interface*
CWT	*Continuous Wavelet Transform*
DB	*Distance Bounding*
EKF	*Extended Kalman Filter*
FFT	*Fast Fourier Transform*
GB	*Gradient Boosting*
GPS	*Global Positioning System*
HC-SLE	*Highest Context-Based Step Length Estimation*
INS	*Inertial Navigation System*
KF	*Kalman Filter*
KNN	*K-Nearest Neighbors*
MEMS	*Microelectromechanical systems*
PCA	*Principal Component Analysis*
PDR	*Pedestrian Dead Reckoning*
PKES	*Passive Keyless Entry and Start*
RBF	*Radial Basis Function*
RFT	*Reference Frame Transformation*
RKE	*Remote Keyless Entry*
RMSE	*Root-Mean-Square Error*
RVM	*Relevance Vector Machine*
SHS	*Step-and-Heading System*
STFT	*Short Time Fourier Transform*
SVM	*Support Vector Machine*
WC-SLE	*Weighted Context-Based Step Length Estimation*
XGBoost	*Extreme Gradient Boosting*

SUMÁRIO

1 INTRODUÇÃO

A primeira forma de se entrar em um veículo foi através da chave mecânica. Sua principal função era abrir as portas e permitir que o motor do carro fosse ligado (GARCIA et al., 2016). Entretanto as chave puramente mecânicas são as mais vunáveis que existem, motivo este que levou a sua extinção. Eram facilmente copiadas, bastava-se ter acesso a chave original, e eram bastante veneráveis a técnicas de *lock-picking* (Figura 1).

Figura 1 – Utilização de técnica de *lock-picking* para abrir porta de um veículo.

Fonte: (ME, 2018)

Em resposta surgiram as chaves mecânicas com *immobilizer*. As chaves agora possuíam um circuito eletrônico dentro do seu envólucro plástico, que escondia um segredo (MASON, 2012). Desta forma, eram capazes de autenticar-se com os veículos impedindo que os mesmos fossem ligados por ligação direta. Apenas a chave real poderia ligar o motor do veículo. Com esta técnica o número de roubos de veículos caiu em 40% (OURS; VOLLAARD, 2016). Uma das grandes vantagens do *immobilizer* é que o usuário não precisar ligar nada, ele funciona de forma totalmente transparente.

Com o passar dos anos e com a evolução tecnológica, na década de 80 surgiram os sistemas *Remote Keyless Entry* (RKE) sistemas capazes de abrir e travar o veículo a distância, apenas apertando um botão na chave do veículo. A Figura 2 mostra um exemplo de chave com sistema RKE. Os sistemas RKE possuem um circuito de rádio frequência embutido na chave que se comunica com outro circuito de rádio frequência localizado no veículo (MASON, 2012). Quando o usuário pressiona o botão, uma mensagem é enviada para o veículo requisitando que o mesmo trave ou destrave. A distância de comunicação varia de acordo com a tecnologia.

Figura 2 – Chave com sistema *Remote Keyless Entry* (RKE)

Hoje em dia, é comum ver carros equipados com sistemas *Passive Keyless Entry and Start* (PKES). Estes sistemas podem ser vistos como uma evolução dos sistemas RKE. Seu funcionamento é semelhante do ponto de vista tecnológico, ambos utilizam circuitos de rádio frequência para comunicação, entretanto não existe mais a necessidade de se apertar botões para fazer o travamento e o destravamento do veículo, e ainda não é mais necessário colocar a chave na ignição do veículo para ligar o mesmo. Tudo ocorre de forma passiva (GARCIA et al., 2016), basta estar com a chave no bolso e se aproximar do veículo para abri-lo, afastar-se para travá-lo e estar com a chave dentro do interior do veículo para ligar o motor. Nestes sistemas chaves mecânicas funcionam apenas como um sistema reserva.

Atualmente, já é possível encontrar carros de luxo, como por exemplo, o Tesla Model 3, capazes de utilizar o *smartphone* como sistema PKES. Nestes casos, a comunicação é feita através do Bluetooth, basta o *smartphone* ficar com o Bluetooth ligado continuamente que o mesmo irá funcionar como chave passiva (MOTORS, 2019).

1.1 ATAQUE DE *RELAY*

Sistemas PKES permitem a abertura passiva por algum mecanismo de proximidade, normalmente medindo-se a potência de sinal da chave no veículo (RANGANATHAN; CAPKUN, 2018). Desta forma se, a chave estiver a uma distância pré determinada (por exemplo, 2m), o veículo permite a sua abertura.

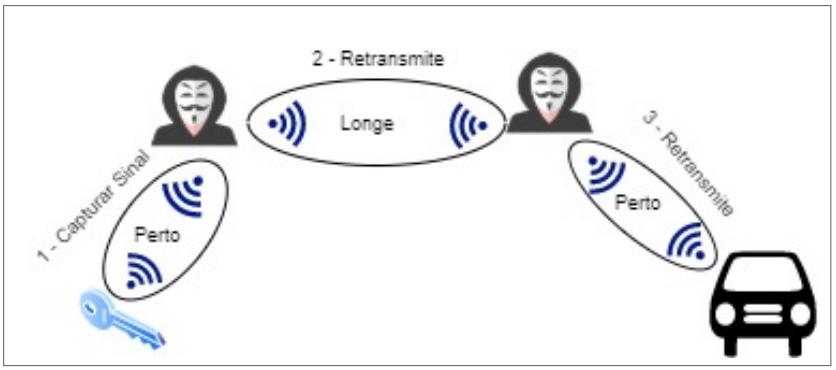
Toda a facilidade e conforto fornecidos pelos sistemas PKES tem uma consequência, sistemas PKES são vulneráveis a ataques *Relay*. Por ser completamente pas-

sivo, ou seja, ausência de interação com o usuário, não é possível para o usuário descobrir quando o seu sistema está sendo utilizado. E por isso, ataques de *Relay* são bastante efetivos.

Ataques *Relay* funcionam da seguinte forma. Dois atacantes, A e B, fazem uma ponte entre as partes autênticas, veículo e chave. O atacante A fica próximo da chave e o atacante B próximo do veículo. O atacante A passa a copiar as mensagens que estão sendo transmitidas pela chave e enviá-las para o atacante B e então o atacante B as envia para o veículo. Tudo que o veículo vê são apenas mensagens válidas da chave sendo transmitidas a uma curta distância. Desta forma, o veículo permite que o mesmo seja aberto (FRANCILLON; DANEV; CAPKUN, 2011).

De forma resumida, o ataque faz com que as partes pareçam estar mais perto do que realmente estão, uma vez que a potência de sinal medida pelo veículo está dentro dos limites para se detectar a chave como próxima. Como não existe nenhuma interação com o usuário (sistema puramente passivo), o mesmo não fica ciente de que o veículo foi aberto. O esquema do ataque pode ser visto na Figura 3.

Figura 3 – Ataque de *Relay*



1.2 *DISTANCE BOUNDING*

Uma das formas existentes para se defender de ataques de *Relay* e implementando-se *Distance Bounding* (DB) na camada física. Protocolos de DB tem como objetivo medir a distância entre as partes comunicantes através do tempo de resposta de algumas mensagens de desafio. Sabendo-se a velocidade com que as ondas de rádio frequência se movem e o tempo necessário para processar estas mensagens de desafio, pode-se inferir um limite superior para a distância entre as partes comunicantes, em sistemas PKES, entre a chave e o veículo.

No caso de um ataque de *Relay* os atrasos gerados pela comunicação entre os dois atacantes fariam com que a distância inferida fosse muito grande desta forma o veículo não seria aberto.

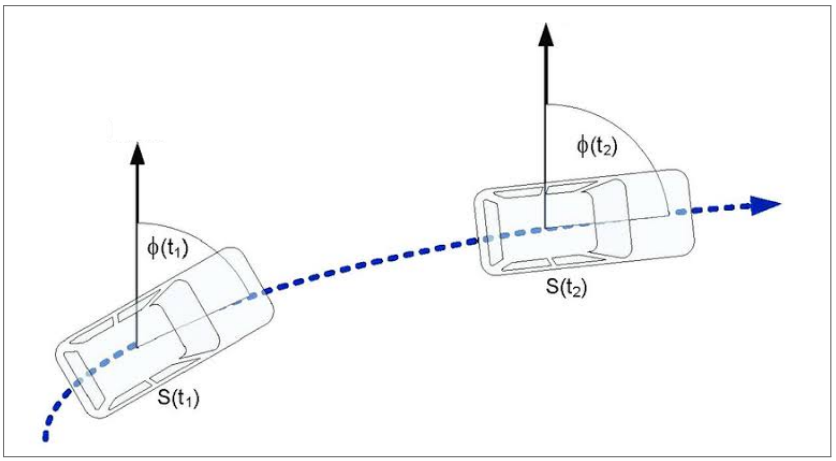
No entanto, protocolos de DB são sensíveis ao menor atraso que possa ocorrer no processamento destas mensagens de desafio, nas partes autênticas. Isso deve-se ao fato das ondas de rádio frequência se moverem a aproximadamente 299.792.458 m/s (velocidade da luz), então um atraso da ordem de 10ns causaria um erro na distância inferida na ordem de 1,5m (RANGANATHAN; CAPKUN, 2018). Imaginando-se o cenário do *smartphone* como chave PKES, um atraso na ordem de 1ms por parte da aplicação é situação totalmente plausível de vir a acorrer, e um atraso desta ordem levaria a milhares de metros de erro na inferência de sua distância, o que causaria negação de serviço ao usuário.

1.3 *DEAD RECKONING*

Uma técnica bastante utilizada para navegação em embarcações é a técnica *Dead Reckoning*, que é capaz de estimar a localização do usuário apenas com alguns sensores e um ponto de origem. Os principais instrumentos para este tipo de navegação são a bússola e o velocímetro que servem respectivamente para estimar a direção e a distância percorrida. Sabendo-se sua posição inicial e para onde se está indo (direção, sentido e distância percorrida), é possível determinar sua nova posição. O processo básico pode ser visto na Figura 4.

Existem diversos trabalhos que utilizam os sensores inerciais dos (magnetômetro, giroscópio e acelerômetro) *smartphones* para se fazer *Dead Reckoning* (SHIN et al., 2016; HUANG et al., 2016; KLEIN; SOLAZ; OHAYON, 2018; WANG et al., 2018), alguns com localização 2D outros 3D. Existem também variações quanto a liberdade de posições, alguns trabalhos com posições fixas (MARTINELLI et al., 2018) outros com posições variadas (SHIN et al., 2016).

Figura 4 – Processo de localização baseado em *Dead Reckoning*. $S(t_1)$ e $S(t_2)$ são as posições em dois instantes de tempo consecutivos. $\phi(t_1)$ e $\phi(t_2)$ são os ângulos desses mesmos instantes.



1.4 OBJETIVOS

1.4.1 Objetivo Geral

O objetivo principal deste trabalho é propor um sistema de localização 2D, utilizando um *smartphone*, que seja capaz de servir como sistema PKES inume a ataques de *Relay*.

1.4.2 Objetivos Específicos

Este trabalho possui os seguintes objetivos específicos:

- Desenvolver um algoritmo de *Dead Reckoning* para efetuar localização 2D do usuário em tempo real, otimizado para sistemas PKES.
- Construir um aplicativo para sistema Android capaz de coletar e armazenar em memória flash leituras dos sensores inerciais (magnetômetro, giroscópio, acelerômetro e *Rotation Vector*).
- Construir uma base de dados pública a partir do aplicativo Android, assim permitindo maior facilidade na replicação dos resultados obtidos por este trabalho.

1.5 ORGANIZAÇÃO DA DISSERTAÇÃO

O presente trabalho está organizado da seguinte forma:

- **Capítulo 2 - Fundamentação Teórica**: são apresentados os conceitos básicos para entendimento do trabalho.
- **Capítulo 3 - Trabalhos Relacionados**: são apresentados trabalhos relacionados a sistemas de *Pedestrian Dead Reckoning* (PDR).
- **Capítulo 4 - Método Proposto**: é detalhado o sistema proposto neste trabalho.
- **Capítulo 5 - Base de dados**: é feita a descrição dos dados gerados para os experimentos.
- **Capítulo 6 - Experimentos e Discussões**: são detalhados os experimentos feitos utilizando a base de dados descrita no Capítulo 5
- **Capítulo 7 - Conclusão**: são apresentados as conclusões gerais, dificuldades, contribuições e trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste Capítulo serão abordados temas que servem de base teórica para os demais Capítulos. Na Seção 2.1 será explicado o funcionamento dos sensores utilizados como fonte de dados do algoritmo proposto (Capítulo 4). Na Seção 2.2 será explicada a técnica de localização utilizada pelo algoritmo proposto. A Seção 2.3 detalha cálculos necessários em algumas etapas do algoritmo proposto. Por fim, na Seção 2.4, será detalhada a principal técnica de aprendizagem de máquina utilizada como base do algoritmo proposto.

2.1 SENSORES

2.1.1 Giroscópio

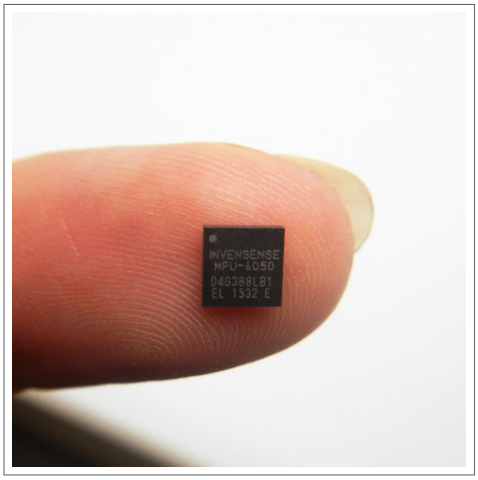
Giroscópios são dispositivos utilizados para medir a orientação e/ou a velocidade angular de um determinado dispositivo ou veículo (aeronaves, carros, navios). É composto por um disco giratório em que o eixo de rotação possui liberdade para apresentar qualquer tipo de orientação, como por ser visto na Figura 5.

Figura 5 – Giroscópio.

Existe mais de um princípio de funcionamento para giroscópios, os principais são o rotatório (Figura 5), ótico e o de estruturas vibrantes. Os giroscópios de estruturas vibrantes são encontrados comumente em circuitos integrados através da tecnologia conhecida como *Microelectromechanical systems* (MEMS). Esta tecnologia permite a criação de partes móveis em escala microscópica, que podem ser empregada nos circuitos integrados, que são essenciais para dispositivos *mobile* como *smartphones* e *smartbands*. Na Figura 6 podemos ver um exemplo de circuito integrado que possui um giroscópio MEMS.

Um giroscópio MEMS pode ser classificado como de 1, 2 ou 3 eixos isto representa em quantos eixos ele consegue medir velocidades angulares, atualmente a maioria dos circuitos integrados de giroscópios são de 3 eixos. Em *smartphones* Android, giroscópios, quando presentes, são de 3 eixos como está descrito na *Application Programming Interface* (API) do Android (ANDROID, 2019).

Figura 6 – MPU-6050 circuito integrado que possui acelerômetro e giroscópio, ambos de 3 eixos.



2.1.2 Acelerômetro

Acelerômetros são dispositivos capazes de medir a aceleração inercial. Assim como os giroscópios, podem ser classificados como de 1, 2 ou 3 eixos. Acelerômetros possuem diversas aplicações entre as principais tem-se, como sensor capaz de medir velocidade e posição (HUA; SHEN; ZHONG, 2017), sensor de inclinação medindo a aceleração da gravidade (HUNG et al., 2017) e como detectores de queda (LEONE; RESCIO; SICILIANO, 2013).

Existe mais de um princípio de funcionamento para acelerômetros os mais comuns são baseados em sensores capacitivos e baseados em efeito piezoelétrico. Independente do principio todos se baseiam pelo movimento de uma massa de prova, e através da força exercida por essa massa de prova calcula-se a aceleração.

Para exemplificar o porquê de uma massa de prova o modelo que será considerado é o seguinte, um cubo com uma pequena esfera dentro deste cubo conforme Figura 7.

Neste modelo cada uma das faces do cubo ($X+$, $Y+$, $Z+$, $X-$, $Y-$ e $Z+$) e capaz de medir uma força aplicada nela, e a partir desta força, conhecendo-se a massa da esfera, pode-se inferir a aceleração. Na Figura 8 pode se ver o efeito de uma força F com aceleração de $1g$ ($1g = 9.8m/s^2$) que é aplicada no cubo. Por inercia a esfera esbarra na parede $X-$ e dando uma leitura de $-1g$.

Figura 7 – Modelo do acelerômetro.

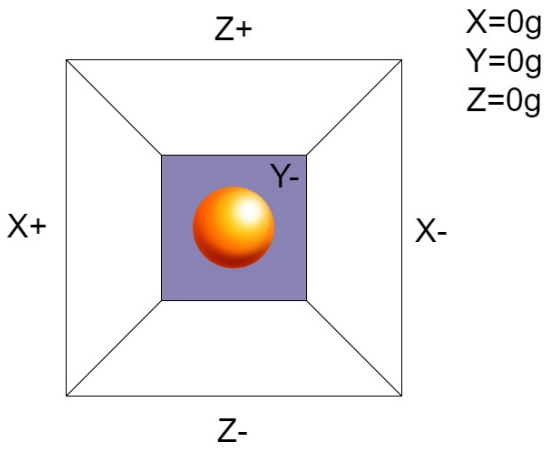
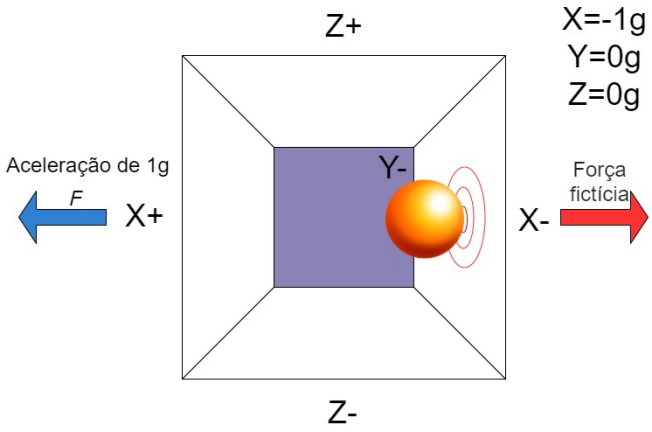
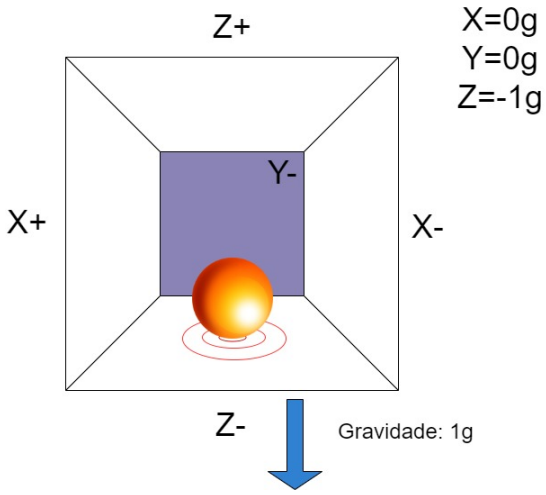


Figura 8 – Modelo do acelerômetro com força (aceleração de 1g) sendo aplicada na direção $X+$.



Quando o sistema está em repouso, tem-se a situação representada pela Figura 9, onde a esfera está unicamente sobre o efeito da força peso. Assim, sempre temos que considerar que o sistema também está sobre efeito da força peso, então uma leitura com magnitude $1g$ não indica, necessariamente, um movimento.

Figura 9 – Modelo do acelerômetro em repouso sobre efeito da força peso.

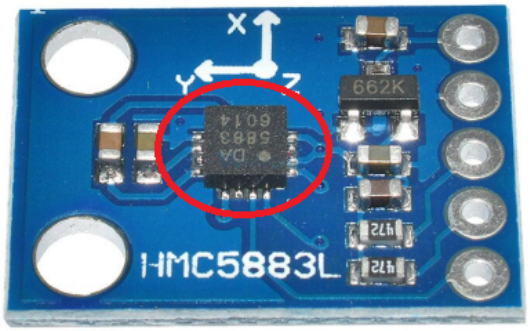


2.1.3 Magnetômetro

Magnetômetros são dispositivos capazes de medir a direção e magnitude de campos magnéticos. São muito empregados como bússolas digitais em componentes eletrônicos, para esta funcionalidade basta detectar o campo magnético terrestre.

Assim como os giroscópios e acelerômetros, podem ser miniaturizados em circuitos integrados, um exemplo de circuito integrado que é um magnetômetro pode ser visto na Figura 10.

Figura 10 – Circuito integrado HMC5883L, magnetômetro de 3 eixos (circulado em vermelho).



Magnetômetros em circuitos integrados normalmente medem os campos magnéticos através de efeitos da força de Lorentz, medindo alguma variação de tensão elétrica.

2.2 *PEDESTRIAN DEAD RECKONING*

Quando se emprega alguma técnica de *Dead Reckoning* para localização de pessoas, esta passa a ser definida como *Pedestrian Dead Reckoning* (PDR). Os dispo-

sitivos normalmente utilizados para esta função são *smartphones* ou dispositivos vestíveis, como, por exemplo, *smartwatches* e *smartbands*.

As técnicas de PDR se dividem em duas categorias: *Step-and-Heading System* (SHS) e *Inertial Navigation System* (INS) (HARLE, 2013). Sistemas INS são caracterizados por estimar a posição continuamente a qualquer instante de tempo. Sistemas INS podem ser utilizados tanto para cálculo de posição em 2 dimensões quanto em 3 dimensões. Um exemplo simples seria integrar as leituras de aceleração do giroscópio para obter a velocidade e depois integrar novamente para obter o deslocamento.

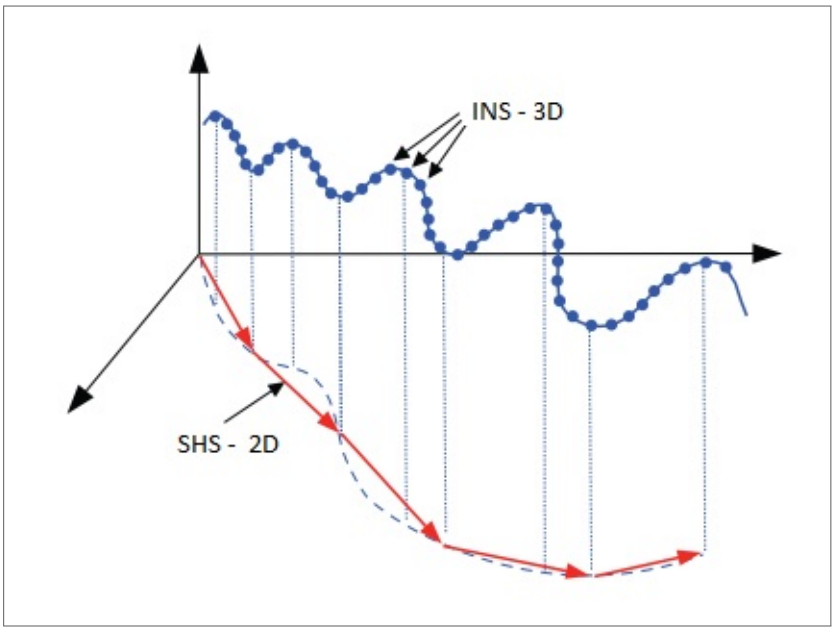
Esta tática mesmo aliada a técnicas de filtragem, como, por exemplo, filtro de Kalman, não tem um desempenho bom devido aos erros acumulados com o tempo. Este erros estão na maior parte do tempo ligado a baixa acurácia dos sensores que estão embutidos em *smartphones*. Sistema que utilizam INS só obtém um bom resultado quando utilizado com sensores de alta acurácia como os existentes em aviões e embarcações (HARLE, 2013).

Sistemas do tipo SHS só podem ser utilizados para PDR e para cálculo de posição em 2 dimensões. Sistemas SHS estimam a posição através de cálculo contínuo de vetores (tamanho e direção), onde estes vetores representam passos. Estes sistemas normalmente possuem três etapas:

1. Detecção de passo: Nesta etapa os eventos de passos são detectados.
2. Estimação do tamanho do passo: Estimar o comprimento da passada.
3. Cálculo de direção: Estimar a direção do passo ou a variação na direção.

Uma comparação visual entre as duas técnicas pode ser vista na Figura 11.

Figura 11 – Comparação entre sistemas INS e SHS. As setas vermelhas representam os passos detectados. Os eixos são as coordenadas espaciais.



Fonte: Adaptado de (HARLE, 2013).

2.3 MATRIZ DE ROTAÇÃO

Uma matriz de rotação é uma matriz quadrada que serve para rotacionar um vetor em torno de um eixo de rotação sem modificar sua magnitude. O eixo de rotação e o valor do ângulo (Figura 12) são definidos pelos elementos da matriz. O vetor rotacionado é dado pela multiplicação da matriz de rotação pelo vetor (no formato de um vetor coluna).

Para um sistema cartesiano de três coordenadas (x, y e z), tem-se como exemplo de matriz de rotação R, a seguinte matriz:

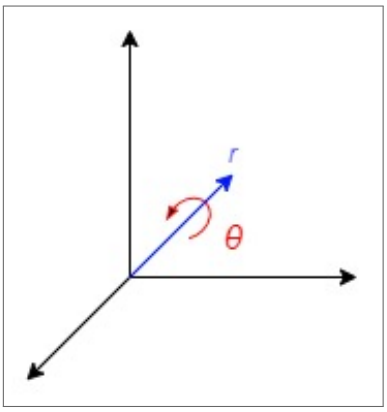
$$R = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

que rotaciona um vetor $[x,\, y,\, z]^T$ ao longo do eixo z por um ângulo θ.

2.3.1 Representação por Quaternião

Quaterniões unitários, também conhecidos como versores, podem prover orientações e rotações em três dimensões assim como as matrizes de rotação. Comparados como matrizes de rotação são mais estáveis numericamente, mais compactos e mais eficientes. Quando quaterniões são utilizados para representar uma rotação são chamados de quaterniões de orientação ou quaterniões de rotação.

Figura 12 – Vetor r e ângulo θ de um rotação. Ambos são necessários para caracterizar uma rotação no espaço.



Uma rotação dada por um ângulo θ em torno de um vetor unitário r pode ser codificada em quaternião de rotação $\hat{q}$ pela seguinte equação:

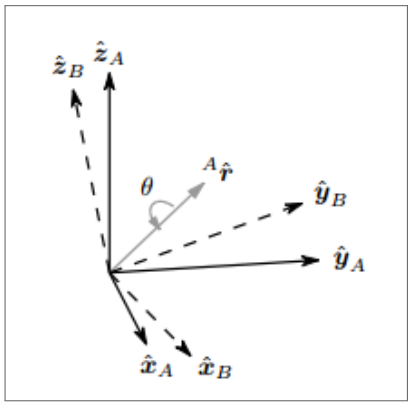
$$\hat{q} = [q_1,\, q_2,\, q_3,\, q_4] = \left[\cos\left(\frac{\theta}{2}\right), -r_x \sin\left(\frac{\theta}{2}\right), -r_y \sin\left(\frac{\theta}{2}\right), -r_z \sin\left(\frac{\theta}{2}\right)\right] \tag{2.2}$$

onde r_x, r_y e r_z são as coordenadas do vetor r.

2.3.2 Rotação Através de Um Quaternião

Quaterniões tanto podem representar uma rotação quanto uma orientação uma vez que uma orientação é uma rotação a partir de um sistema de coordenadas específico. Uma orientação B relativa a um sistema de coordenadas A pode ser alcançada por meio de uma rotação de um ângulo θ em torno de um vetor $A_{\hat{r}}$ definido no sistema de coordenadas A. Isso é representado na Figura 13. Para representar esta

Figura 13 – Orientação B adquirida a partir do sistema de coordenadas A.



orientação é utiliza a seguinte notação ${}^A_B\hat{q}$ que indica uma rotação de A para B, ou seja, a mudança de orientação.

O conjugado do quaternião denotado por $\hat{q}^*$ pode ser utilizado para inverter as rotações, passar de um sistema de coordenadas B para uma orientação A referente

ao sistema de coordenadas B, por exemplo ${}^{B}_{A}\hat{q}$ é o conjugado de ${}^{A}_{B}\hat{q}$. O conjugado de um quaternião é definido como

$$ {}^{A}_{B}\hat{q}^{*} = {}^{B}_{A}\hat{q} = [q_1, -q_2, -q_3, -q_4] \tag{2.3} $$

.

O produto de quaterniões denotado por $\otimes$ pode ser utilizado para compor rotações. Por exemplo, duas rotações ${}^{A}_{B}\hat{q}$ e ${}^{B}_{C}\hat{q}$ podem ser compostas em uma única rotação ${}^{A}_{C}\hat{q}$. A orientação ${}^{A}_{C}\hat{q}$ pode ser definida como

$$ {}^{A}_{C}\hat{q} = {}^{B}_{C}\hat{q} \otimes {}^{A}_{B}\hat{q} \tag{2.4} $$

em que o produto $a \otimes b$ é definido como

$$ \begin{gathered} a = [a_1, a_2, a_3, a_4] \\ b = [b_1, b_2, b_3, b_4] \\ a \otimes b = \begin{bmatrix} a_1b_1 - a_2b_2 - a_3b_3 - a_4b_4 \\ a_1b_2 + a_2b_1 + a_3b_4 - a_4b_3 \\ a_1b_3 - a_2b_4 + a_3b_1 + a_4b_2 \\ a_1b_4 + a_2b_3 - a_3b_2 + a_4b_1 \end{bmatrix}^{T} \end{gathered} \tag{2.5} $$

. O operador $\otimes$ não é comutativo, logo $a \otimes b \neq b \otimes a$.

Um vetor tridimensional ${}^{A}v$ no sistema de coordenadas A pode ser rotacionado para uma orientação B gerando o vetor ${}^{B}v$ a partir de um quaternião ${}^{A}_{B}\hat{q}$. A rotação de ${}^{A}v$ para a orientação B é definida como

$$ {}^{B}v = {}^{A}_{B}\hat{q} \otimes {}^{A}v \otimes {}^{A}_{B}\hat{q}^{*} \tag{2.6} $$

em que o vetor tridimensional $v = [v_x, v_y, v_z]$ deve ser expresso como $[0, v_x, v_y, v_z]$ para poder operar com $\otimes$.

2.4 APRENDIZAGEM DE MÁQUINA

Aprendizagem de máquina ou aprendizado de máquina é um campo científico que estuda algoritmos e modelos estatísticos para executar uma determinada tarefa sem receber instruções específicas de como resolve-la. Para isto, recai em descobrir padrões e fazer inferências. Aprendizagem de máquina é vista como um subconjunto da inteligência artificial.

Algoritmos de aprendizagem de máquina constroem o seu "conhecimento"através de dados já conhecidos sobre o problema. Estes dados são normalmente chamados

de conjunto de treino, e servem de base para extrair os padrões necessários para a resolução do problema.

Aprendizagem de máquina é muito relacionada com computação uma vez que foca em fazer o processamento dos dados utilizando computadores. Aprendizagem de máquina é empregada em diversas tarefas, quais sejam, sistemas de recomendações, filtragem de spam em e-mails, detecção de objetos em imagens, problemas em que é muito difícil ou inviável de se desenvolver de fato um algoritmo para executar a tarefa.

Aprendizagem de máquina possui três grandes divisões que depende do tipo de dados de entrada e saída, e a natureza do problema a ser resolvido:

- Aprendizado supervisionado: um modelo é construído em cima de dados de treino em que se possui ambos os dados de entrada desejados quanto os dados de saída desejados de cada uma das entradas. Os dados de treino são formados por conjunto de instâncias.

  Cada instância de treino pode ter mais de uma entrada e mais de uma saída desejada. O aprendizado supervisionado é dividido em problemas de classificação e regressão.

- Aprendizado não supervisionado: dado um conjunto de dados, apenas entradas, nenhuma saída, um algoritmo tem que encontrar alguma estrutura nesses dados. Para tanto, estes algoritmos aprendem sobre dados que não foram classificados ou categorizados.

  Diferentemente do aprendizado supervisionado, que durante o treino pode-se ter um *feedback* de quão boa a solução está, no aprendizado não supervisionado isto não é possível, a forma encontrada para "aprender", é tentando identificar semelhanças entre as instâncias de treino.

- Aprendizado por reforço: é uma área da aprendizagem de máquina que está preocupada em como um agente toma ações em um determinado ambiente, tentando maximizar uma determinada premiação. Aprendizado por reforço é muito utilizada em sistema autônomos e em como aprender a jogar um jogo com um humano.

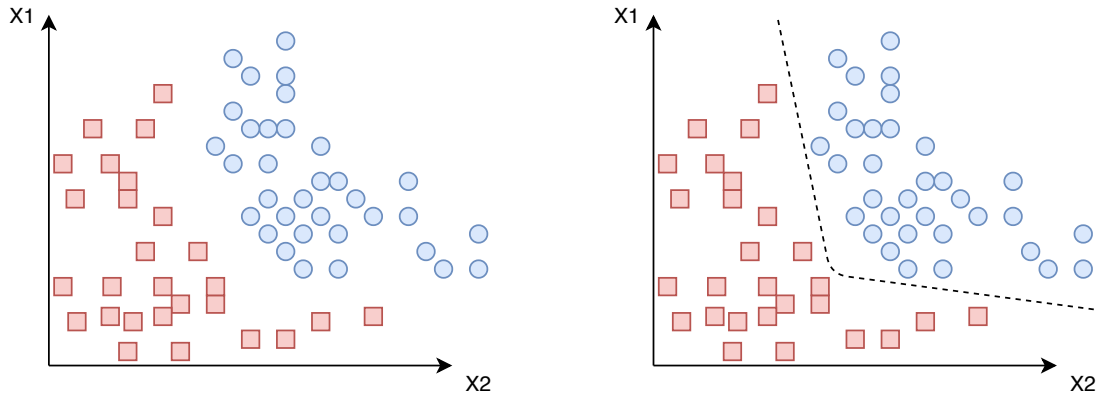
2.4.1 Problemas de Classificação

Classificação é o problema de se identificar a que conjunto de categorias uma nova observação (instância de classificação) pertence, baseado em um conjunto de treinamento, previamente conhecido, em que as categorias das instâncias são conhecidas.

Como exemplo de problema de classificação temos, análise de crédito, identificar pessoas como bons pagadores e maus pagadores. Um algoritmo que implementa uma classificação é conhecido como classificador. Uma de suas funções é encontrar superfícies de decisão, que separaram duas categorias a partir das suas características.

Um exemplo de superfície de decisão encontrada por um classificador, para o problema ilustrado na Figura 14a, pode ser vista na Figura 14b, em que $X1$ e $X2$ são as características de cada instância de classificação.

Figura 14 – Problema de classificação com 2 classes.



(a) Conjunto de dados.

(b) Superfície de decisão.

2.4.2 Problemas de Regressão

Modelos de regressão são utilizados para prever valores contínuos. Por exemplo, prever o preço de uma casa dadas algumas características como tamanho, número de quartos, região em que se encontra etc.

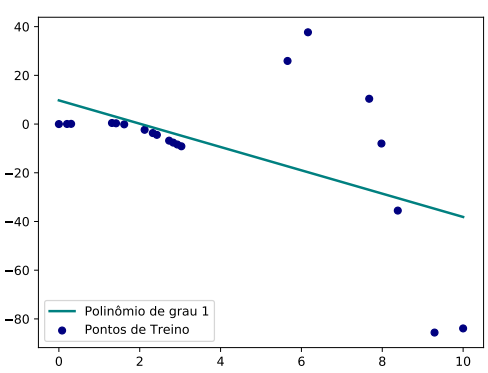
Um algoritmo que implementa uma regressão é conhecido como regressor. A exemplo de regressores temos a regressão linear que tenta "aprender"a função que rege os dados utilizando uma reta (Figura 15a), e temos a regressão polinomial que tenta resolver o problema utilizando polinômios de graus maiores que 1 (Figura 15b).

2.4.3 SVM

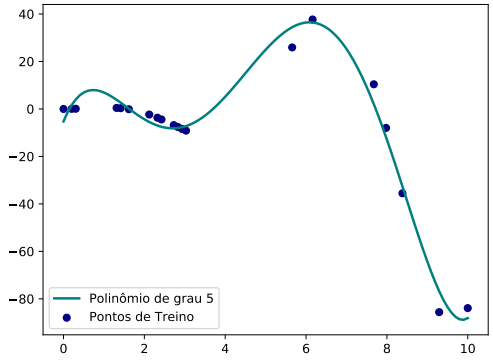
Support Vector Machine (SVM) (BOSER; GUYON; VAPNIK, 1992) são modelos de aprendizado supervisionados utilizados na resolução de problemas de classificação e regressão, lineares e não lineares (MITCHELL, 1997).

SVMs têm sido empregadas em diversas áreas, dentre elas, detecção de faces (Osuna; Freund; Girosit, 1997), categorização de textos (JOACHIMS, 1998), classifica-

Figura 15 – Problema de regressão.



(a) Regressão Linear.



(b) Regressão polinomial com polinômio de grau 5.

ção de imagens (Foody; Mathur, 2004), reconhecimento de caracteres manuscritos (AHAMED; ALAM; ISLAM, 2019; KARIM; SAADOON; ABDULLAH, 2019), etc.

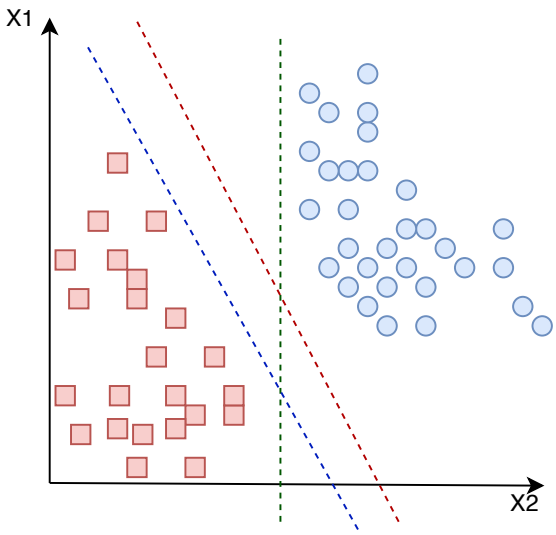
O funcionamento da SVM vem da ideia de maximizar a distância da região de decisão para cada uma das classes envolvidas. Na Figura 16a pode-se ver que todos os três hiperplanos (linhas pontilhadas em vermelho, azul e verde) separam as duas classes presentes (quadrados vermelhos e círculos azuis). Entretanto, os hiperplanos verde e azul estão muito próximos das classes, círculo e quadrado, respectivamente.

O hiperplano vermelho por ter uma distância maior tanto dos quadrados quanto dos círculos fornece uma região de separação com capacidade de generalização maior que os outros hiperplanos.

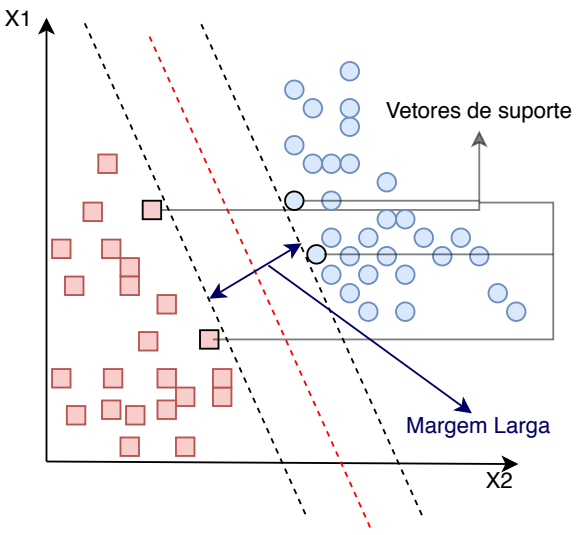
Para maximizar a distância da região de decisão para cada uma das classes envolvidas, é necessário encontrar os vetores de suportes, que são os pontos de cada classe mais próximos do hiperplano (Figura 16b). Com estes pontos é possível calcular as margens do classificador, a partir desse cálculo, pode ser encontrar o hiperplano que maximiza a margem.

Figura 16 – Regiões de decisão de uma SVM.

(a) Hiperplanos de separação.



(b) Vetores de suporte de cada classe.



3 TRABALHOS RELACIONADOS

Os trabalhos relacionados ao sistema proposto são apresentados neste Capítulo. Primeiramente será feito uma apanhado geral dos trabalhos na Seção 3.1 e nas demais Seções serão detalhados os trabalhos mais atuais e relevantes dentre os citados neste Capítulo.

3.1 SISTEMAS DE PDR

A maioria dos trabalhos anteriores a este empregam alguma técnica de aprendizagem de máquina ou reconhecimento de padrões no intuito de aumentar a acurácia geral do sistema. Diferentes pesquisadores tentaram combinações variadas de posições do *smartphone* (bolso, mão etc) assim como diferentes padrões de movimentos (andando, parado, correndo, subindo escadas etc) (SHIN et al., 2016; HUANG et al., 2016; KLEIN; SOLAZ; OHAYON, 2018; WANG et al., 2018; MARTINELLI et al., 2018). Neste trabalho serão abordados apenas algoritmos de PDR do tipo SHS por serem os que apresentam melhores resultados (HARLE, 2013).

Como visto na Seção 2.2, algoritmos de PDR do tipo SHS possuem três principais etapas: 1) Detecção de passo, 2) Estimação do tamanho do passo e 3) Cálculo de direção (*heading*).

Para a etapa de detecção de passo a maioria dos trabalhos utiliza técnicas baseadas em detecção de picos, definidos através de limiares (SHIN et al., 2012; PRATAMA; WIDYAWAN; HIDAYAT, 2012; MODER et al., 2015; ELHOUSHI et al., 2016; SHIN et al., 2016; HUANG et al., 2016; KLEIN; SOLAZ; OHAYON, 2018; WANG et al., 2018). Outras técnicas para detecção de passos também são empregadas como a autocorrelação (HUANG et al., 2016) e a análise de sinais utilizando *Continuous Wavelet Transform* (MARTINELLI et al., 2018).

Classificadores normalmente são empregados na etapa de detecção de passo para ajustar os limiares de detecção de pico (MODER et al., 2015; SHIN et al., 2016; WANG et al., 2018).

Na etapa de estimação do tamanho do passo, vários trabalhos utilizam combinações lineares de algumas métricas, como, por exemplo, frequência de passos e variância do acelerômetro (SHIN et al., 2016; WANG et al., 2018; MARTINELLI et al., 2018). Outros se baseiam em equações não lineares na diferença da leituras máxima e miníma do acelerômetro (PRATAMA; WIDYAWAN; HIDAYAT, 2012; KLEIN; SOLAZ; OHAYON, 2018; KUANG; NIU; CHEN, 2018; WEINBERG, 2002). Alguns trabalhos também utilizam estudos sobre a forma como as pessoas andam e fazem relações com

variações de algumas variáveis, como, por exemplo, altura do torso (HUANG et al., 2016).

Na etapa de estimação do tamanho do passo, os classificadores são empregados, para selecionar as equações que são utilizadas dentro de um conjunto predefinido (ELHOUSHI et al., 2016; MODER et al., 2015) ou então para variar parâmetros de equação (SHIN et al., 2016; WANG et al., 2018; MARTINELLI et al., 2018).

Para a última etapa, cálculo de direção, normalmente são utilizados técnicas de fusão de sensores como *Kalman Filter* (KF) e *Extended Kalman Filter* (EKF) (SHIN et al., 2016; HUANG et al., 2016). Alguns trabalhos também fazem fusão de informações de naturezas, como, por exemplo, fusão de informações de *fingerprint* com a de sensores inerciais (MODER et al., 2015).

3.2 *MOTION RECOGNITION-BASED 3D PEDESTRIAN NAVIGATION SYSTEM USING SMARTPHONE*

O sistema proposto por (SHIN et al., 2016) utiliza os sensores inerciais disponíveis no *smartphone* (acelerômetro, giroscópio e magnetômetro), sensor barimétrico e dados do *Global Positioning System* (GPS) para fazer a localização em três dimensões (3D) do usuário do *smartphone*. Para aumentar a acurácia do algoritmo é considerado que o *smartphone* pode estar em posições diferentes no corpo do usuário. As posições consideradas no trabalho geram seis padrões de movimentos descritos no Quadro 1 e que podem ser vistos na Figura 17.

Quadro 1 – Padrões de movimento considerados por (SHIN et al., 2016).

Padrão de movimento	Descrição
M_1	Parado em pé olhando para o telefone na mão.
M_2	Andando olhando para o telefone.
M_3	Andando com o celular na mão.
M_4	Andando conversando ao telefone.
M_5	Correndo.
M_6	Andando com o telefone no bolso da frente.

Para identificar qual o padrão de movimento está acontecendo, uma *Artificial Neural Network* (ANN) é utilizada. Para treinar o modelo, foram utilizadas 500 amostras de cada classe (M_1, M_2, M_3, M_4, M_5 e M_6). As características utilizada pelo modelo estão descritas no Quadro 2.

O sistema proposto por (SHIN et al., 2016) é do tipo *Step-and-Heading System* (SHS) e engloba as três etapas: Detecção do passo, estimação do tamanho do passo e estimação da direção.

Figura 17 – Padrões de movimento: (a) M_1 (b) M_2 (c) M_3 (d) M_4 (e) M_5 (f) M_6.

Fonte: (SHIN et al., 2016).

Quadro 2 – Características utilizadas pela ANN responsável por identificar o padrão de movimento.

Característica	Descrição
VarAcc	Variância da aceleração.
MeanAccX	Média da aceleração no eixo X.
MeanAccY	Média da aceleração no eixo Y.
MeanAccZ	Média da aceleração no eixo Z.

Detecção do passo

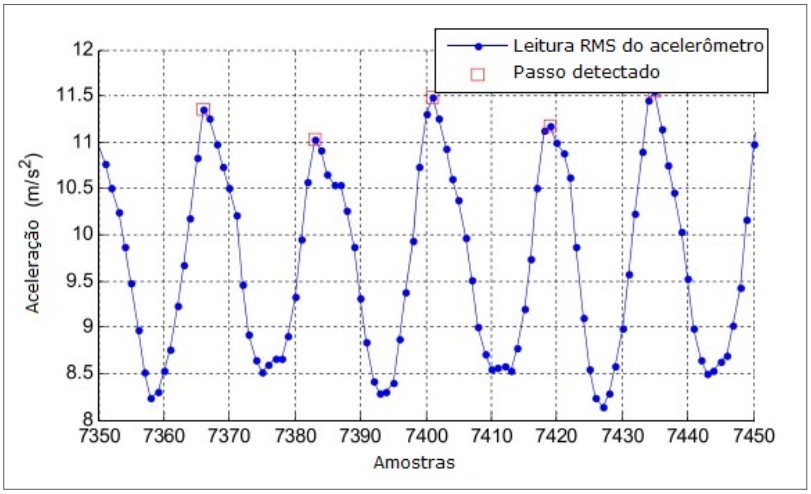
A detecção de passo é feita por meio de detecção de pico de magnitude do acelerômetro. Como a leitura bruta do acelerômetro é muito ruidosa, um filtro de média é utilizado, a detecção de pico então é feita sobre o sinal filtrado do acelerômetro. O processo de detecção de picos pode ser visualizado na Figura 18.

Para cada uma das classes (M_1, M_2, M_3, M_4, M_5 e M_6) existe um valor predefinido de limiar, indicado na Tabela 1.

Padrão	Limiar em m/s^2
M_2	12
M_3	15
M_4	10
M_5	17
M_6	12

Tabela 1 – Limiares para detecção de passo utilizados por (SHIN et al., 2016).

Figura 18 – Detecção de picos sobre dado filtrado do acelerômetro.



Fonte: Adaptado de (SHIN et al., 2016).

Estimação do tamanho do passo

O tamanho do passo (SL_k) é definido como uma combinação linear de três variáveis, a variância do acelerômetro (AV_k), a frequência de passos (WF_k) e a integral do giroscópio (GI_k). A variância do acelerômetro é definida como,

$$AV_k = \frac{1}{n} \sum_{i=t_{k-1}}^{t_k} (a_m - a_i)^2 \tag{3.1}$$

em que, n é número de amostras em um passo, a_m é a média da aceleração no passo, a_i é o valor lido no instante i, t_k é o instante em que o passo foi detectado. A frequência de passos é definida como,

$$WF_k = \frac{1}{n \times \Delta t} \tag{3.2}$$

em que Δt denota a taxa de amostragem. Finalmente, a integral do giroscópio é definida como,

$$GI_k = \frac{1}{n} \sum_{i=t_{k-1}}^{t_k} |g_{yi} \times \Delta t| \tag{3.3}$$

em que g_{yi} é a leitura do giroscópio no eixo y no instante i.

Assim o tamanho do passo é dado como,

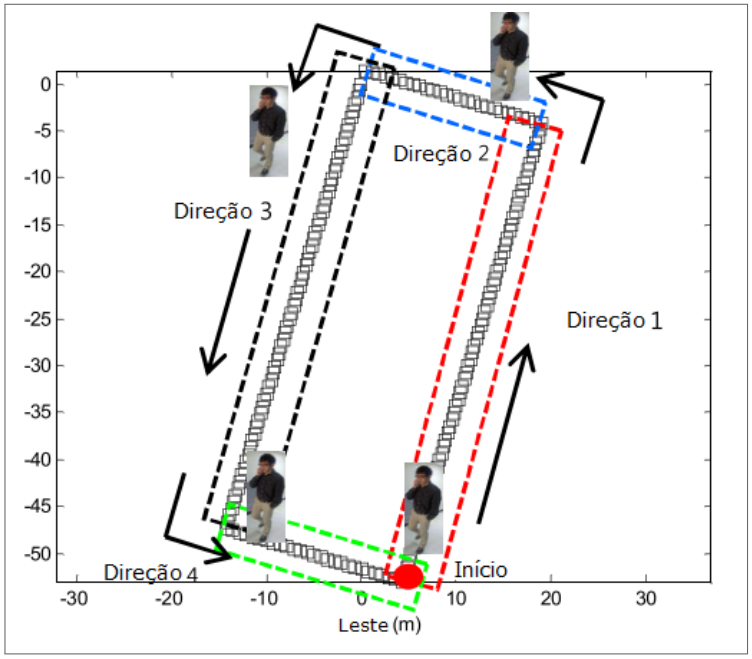
$$SL_k = \alpha \cdot (AV_k - AV_0) + \beta \cdot (WF_k - WF_0) + \gamma \cdot (GI_k - GI_0) + SL_0 \tag{3.4}$$

AV_k, WF_k, GI_k, α, β, γ e SL_0 são parâmetros que devem ser calibrados. α, β e γ são os mesmos independentes da pessoa, os demais devem ser calibrados por pessoa.

Estimação da direção

Para a estimação da direção são empregadas 2 técnicas diferentes dependendo do padrão de movimento. Se o padrão for M_2 é utilizado um filtro de Kalman para definir a direção utilizando como fonte de informação os dados do giroscópio e magnetômetro. Caso seja M_3 ou M_4 ou M_5 ou M_6 é empregado um classificador que diz qual é a direção dentro de 4 possíveis (Figura 19).

Figura 19 – As quatro possíveis direções para o classificador de direção.



Fonte: Adaptado de (SHIN et al., 2016).

O classificador utilizado foi uma ANN para cada um dos padrões M_3, M_4, M_5 e M_6, com entradas definidas no Quadro 3.

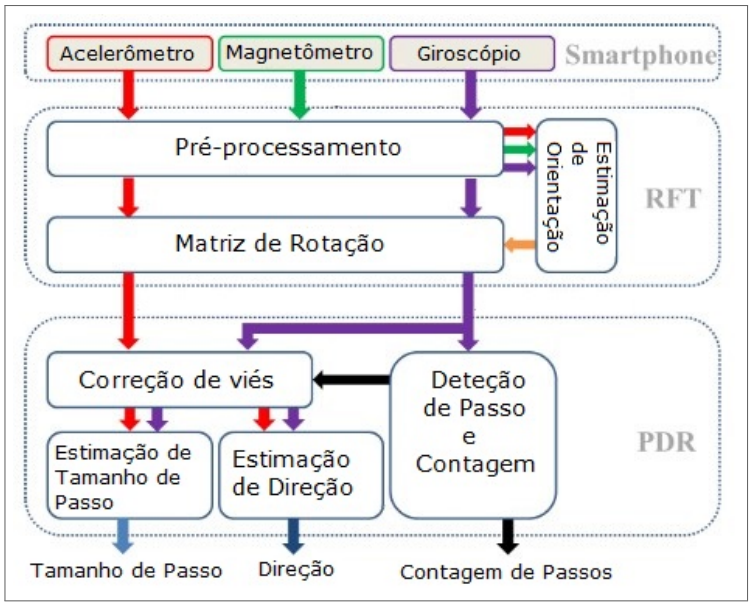
Quadro 3 – Características utilizadas pela ANN responsável por identificar a direção.

Característica	Descrição
YawSin	Valor da componente X do *Yaw* (API Android).
YawCos	Valor da componente Y do *Yaw* (API Android).
MeanMagX	Média do magnetômetro no eixo X.
MeanMagY	Média do magnetômetro no eixo Y.
MeanMagZ	Média do magnetômetro no eixo Z.

3.3 *EXPLOITING CYCLIC FEATURES OF WALKING FOR PEDESTRIAN DEAD RECKONING WITH UNCONSTRAINED SMARTPHONES*

O sistema proposto por (HUANG et al., 2016), considera que o *smartphone* possui acelerômetro, giroscópio e magnetômetro. Para fazer a localização 2D do usuário HUANG et al. propõem um sistema dividido em duas partes, a primeira nomeada de *Reference Frame Transformation* (RFT) e a segunda de PDR. Uma visão geral do sistema pode ser vista na Figura 20.

Figura 20 – Visão geral do sistema proposto por (HUANG et al., 2016).



Fonte: Adaptado de (HUANG et al., 2016).

Inicialmente na etapa RFT, os sinais oriundos do acelerômetro, giroscópio e magnetômetro são processados passando por filtragem para reduzir interferências e vieses nos sinais. Depois, todos alimentam um filtro de Kalman responsável por agregar a informação e calcular a orientação do *smartphone*. A orientação é dada pelo filtro de Kalman como uma matriz de rotação.

A etapa de PDR é um sistema do tipo SHS que possui as três etapas: Detecção do passo, estimação do tamanho do passo e estimação da direção.

Detecção do passo

Para a detecção dos passos, uma janela deslizante é utilizada. Esta janela é responsável por armazenar a velocidade angular no *frame* de referência da Terra no eixo Z. Então é calculada a autocorrelação normalizada e verifica-se se este valor é superior a um limiar Th_2, onde Th_2 é $0.8\,rad/s^2$. Caso este valor seja superior então um passo é contabilizado.

Estimação do tamanho do passo

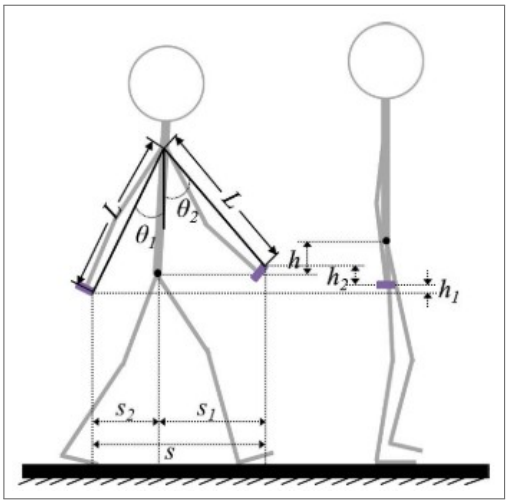
Para o cálculo da tamanho do passo, considera-se que *smartphone* está na mão do usuário, conforme modelo apresentado na Figura 21, em que h_1 e h_2 são dados pelas seguintes equações,

$$h_1 = \int_{T_2}^{T_3} \int_{T_2}^{t} a_z^e(u)dudt \tag{3.5}$$

$$h_2 = \int_{T_3}^{T_4} \int_{T_2}^{t} a_z^e(u)dudt \tag{3.6}$$

em que a_z^e representa a leitura do acelerômetro no eixo z no *frame* terrestre, e T_1, T_2, T_3 e T_4 representam os instantes indicados na Figura 22.

Figura 21 – Modelo de estimação de passo proposto por (HUANG et al., 2016).



Fonte: (HUANG et al., 2016).

Os valores de θ_1 e θ_2 são definidos como

$$\tan^2\theta_1 = \tan^2\left(\int_{T_2}^{T_3} w_x^e(t)dt\right) + \tan^2\left(\int_{T_2}^{T_3} w_y^e(t)dt\right) \tag{3.7}$$

$$\tan^2\theta_2 = \tan^2\left(\int_{T_3}^{T_4} w_x^e(t)dt\right) + \tan^2\left(\int_{T_3}^{T_4} w_y^e(t)dt\right) \tag{3.8}$$

em que w_x^e representa a leitura do giroscópio no *frame* terrestre, e T_1, T_2, T_3 e T_4 representam os instantes indicados na Figura 22.

Assim, pode-se calcular L,

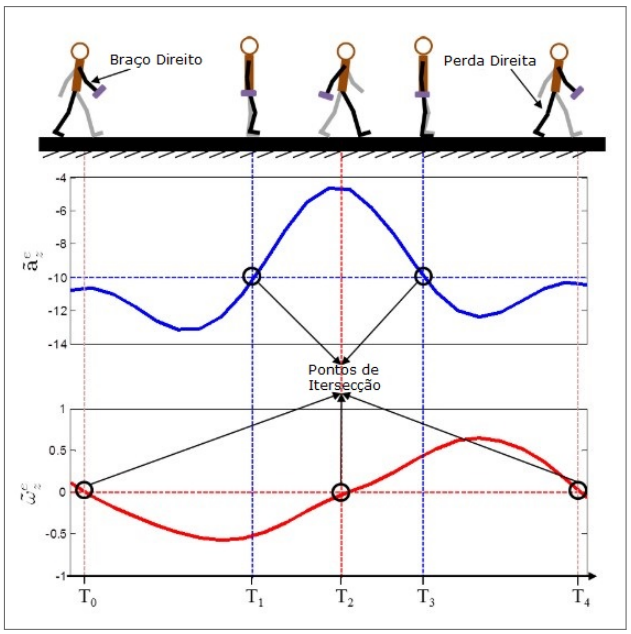
$$L = \frac{h_1 + h_2}{\cos\theta_1 - \cos\theta_2} \tag{3.9}$$

para então calcular o tamanho do passo s definido como,

$$s = 2L\left(sen\,\theta_1 + sen\,\theta_2\right) \tag{3.10}$$

.

Figura 22 – Eventos ao andar com *smartphone* na mão. As linhas pontilhadas azul e vermelha representam respectivamente, a gravidade terrestre e $0\,rad/s$ (HUANG et al., 2016).



Fonte: Adaptado de (HUANG et al., 2016).

Estimação da direção

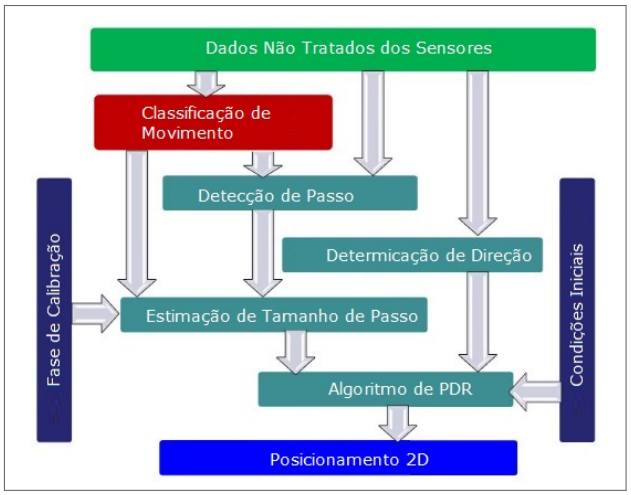
O cálculo da direção é feito através das velocidades angulares de cada ciclo de movimento (dois passos). É intuitivo que se o *smartphone* está na mão, então a direção de movimento do usuário é perpendicular à direção do *smartphone*.

3.4 *PEDESTRIAN DEAD RECKONING WITH SMARTPHONE MODE RECOGNITION*

O sistema proposto por (KLEIN; SOLAZ; OHAYON, 2018) utiliza os sensores inerciais do *smartphone* (acelerômetro, giroscópio e magnetômetro) para fazer localização em duas dimensões (2D). O sistema desenvolvido é um sistema do tipo SHS, no entanto, apenas a etapa de estimação do tamanho do passo é detalhada. As etapas de estimação de direção e de detecção do passo não são detalhadas em nenhuma Seção do artigo.

Para aumentar a acurácia do sistema, é proposta uma arquitetura com reconhecimento de modos de utilização do *smartphone* (em que posições ele pode estar no corpo do usuário). Uma visão geral do sistema com as interações de cada etapa pode ser vista na Figura 23.

Figura 23 – Visão geral do sistema proposto por (KLEIN; SOLAZ; OHAYON, 2018).



Fonte: Adaptado de (KLEIN; SOLAZ; OHAYON, 2018).

Classificação de modos de utilização

Os modos de utilização considerados por (KLEIN; SOLAZ; OHAYON, 2018) foram quatro: 1) bolso, 2) na mão balançando, 3) digitando e 4) falando ao telefone. Além disso, essas quatro classes foram assumidas em 3 condições diferentes de velocidade, com pessoas diferentes com roupas diferentes.

As características utilizadas pelo classificador foram calculadas a partir de uma janela e possuem três naturezas distintas:

- Estatística: média, mediana, desvio padrão, média absoluta da diferença, intervalo interquartil, assimetria, curtose, energia, magnitude, máximo, mínimo e amplitude.
- Domínio do tempo: número de picos, taxa de *median-crossing* e taxa de *g-crossing*.
- Domínio da frequência: Correlação entre o giroscópio e acelerômetro.

De forma resumida são geradas 111 características a partir do giroscópio e acelerômetro, dentre características de tempo, frequência e estatísticas. O classificador utilizado foi um *Gradient Boosting* (GB), utilizando Árvores de Decisão como classificador base.

Estimação do tamanho do passo

Para a estimação do tamanho do passo são utilizadas duas técnicas, a abordagem de Weinberg (WEINBERG, 2002) e a abordagem de Kim (KIM et al., 2004). A abordagem

de Weinberg é baseada na seguinte equação

$$s_k^1 = K_1 \left(\max a_D - \min a_D\right)^{\frac{1}{4}} \tag{3.11}$$

em que K_1 é um ganho que precisa ser calibrado e a_D é a aceleração vertical. A abordagem de Kim é baseada na seguinte equação

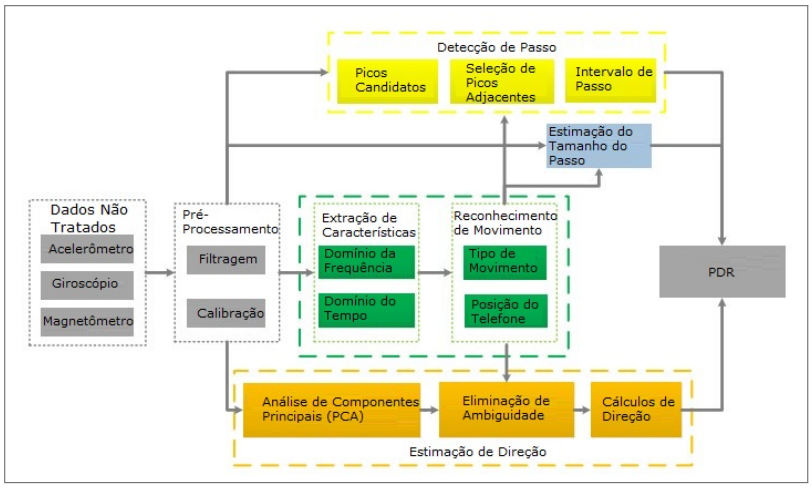
$$s_k^2 = K_2 \left(\sum_{k=1}^{N} |a_D[k]|\right)^{\frac{1}{3}} \tag{3.12}$$

em que N é o número de medições de aceleração durante uma passada e K_2 é um ganho que precisa ser calculado.

3.5 *PEDESTRIAN DEAD RECKONING BASED ON MOTION MODE RECOGNITION USING A SMARTPHONE*

No trabalho desenvolvido por (WANG et al., 2018), é proposto um sistema de PDR do tipo SHS para efetuar localização em duas dimensões (2D). A localização é baseada na utilização dos sensores inerciais do *smartphone* (acelerômetro, giroscópio e magnetômetro). Uma visão geral do sistema proposto por (WANG et al., 2018) pode ser vista na Figura 24.

Figura 24 – Visão geral do sistema proposto por (WANG et al., 2018).



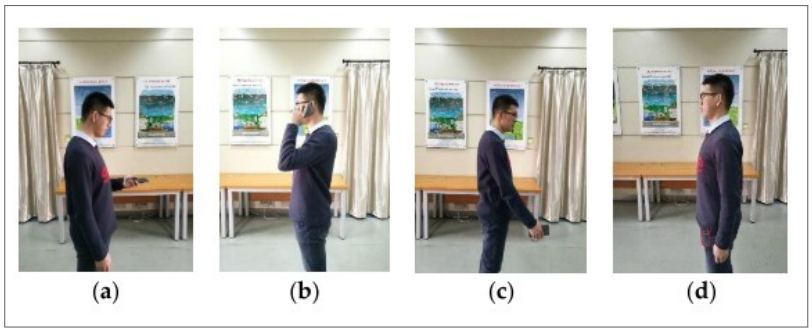
Fonte: Adaptado de (WANG et al., 2018).

Classificação de modos de utilização

Assim como os trabalhos das Seções anteriores, este também utiliza a detecção de modos de utilização para aumentar a acurácia do sistema como um todo. As possíveis posições de utilização do *smartphone* podem ser vistas na Figura 25, quais

sejam, segurando utilizando o *smartphone*, falando, segurando na mão sem utilizá-lo e no bolso. Também foram consideradas quatro possíveis formas de se movimentar: correndo, andando, subindo escadas e descendo escadas. Uma descrição mais detalhada está no Quadro 4.

Figura 25 – Modos de utilização do *smartphone* considerados por (WANG et al., 2018): (a) Segurando utilizando, (b) Falando, (c) Segurando na mão sem utilizar e (d) Bolso.



Fonte: (WANG et al., 2018).

As características utilizadas pelo sistema de reconhecimento de modos de utilização, provém das leituras dos sensores inerciais do *smartphone* acelerômetro, giroscópio e magnetômetro. Porém, antes de serem utilizados passam por um processo de filtragem, como visto na Figura 24.

Quadro 4 – Descrição dos padrões de movimento considerados por (WANG et al., 2018).

Padrão de movimento	Descrição
Segurando utilizando	Neste caso, o *smartphone* está sendo segurado em frente ao corpo de forma estável e a direção do *smartphone* e a mesma direção de movimentação do usuário.
Falando	Neste caso, o usuário está falando ao *smartphone*, que aponta para o lado do usuário.
Segurando na mão sem utilizar	Representa o caso em que o usuário se move segurando o *smartphone* na mão, executando pequenos movimentos de pêndulo.
Bolso	Neste caso, o *smartphone* se encontra no bolso da frente, de forma que o plano do *smartphone* é perpendicular ao chão, quando o usuário está parado.

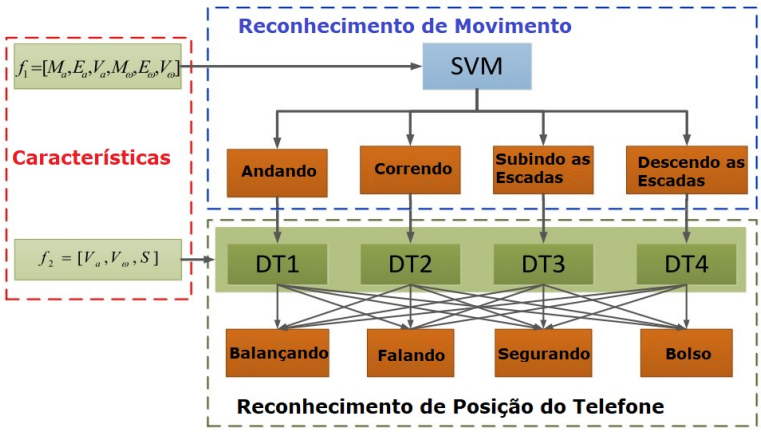
Para gerar as características, é utilizada uma janela deslizante de tamanho 256 com sobreposição de 50%. As características de origem estatística são as seguintes:

- Média aceleração $M_a = [m_{ax}, m_{ay}, m_{az}, m_a]$,
- Média velocidade angular $M_w = [m_{wx}, m_{wy}, m_{wz}, m_w]$,
- Variância aceleração $V_a = [v_{ax}, v_{ay}, v_{az}, v_a]$,
- Variância velocidade angular $V_w = [v_{wx}, v_{wy}, v_{wz}, v_w]$,
- Energia aceleração $E_a = [e_{ax}, e_{ay}, e_{az}, e_a]$,
- Energia velocidade angular $E_w = [e_{wx}, e_{wy}, e_{wz}, e_w]$.

Também são geradas características do domínio da frequência através da *Short Time Fourier Transform* (STFT) definidas como S.

Para fazer a distinção entre correndo, andando, subindo escadas e descendo escadas, foi utilizada uma *Support Vector Machine* (SVM). E para distinguir entre utilizando o *smartphone*, falando, segurando na mão sem utilizá-lo e no bolso foram utilizadas Árvores de Decisão. Um esquema geral pode ser visto na Figura 26.

Figura 26 – Classificador utilizado por (WANG et al., 2018).



Fonte: Adaptado de (WANG et al., 2018).

Detecção do passo

A detecção de passo é feita por um algoritmo de detecção de pico sobre a magnitude das leituras filtradas do acelerômetro. Para cada combinação de posição do *smartphone* e forma de movimento, existe um limiar que tem que ser calibrado. O funcionamento se assemelha ao proposto por (SHIN et al., 2016).

Estimação do tamanho do passo

Para a estimação do tamanho do passo, é utilizada uma combinação linear definida por

$$L = A \cdot f_{stride} + B \cdot \sigma_a^2 + C \tag{3.13}$$

em que L é o tamanho de passo estimado, f_{stride} é a frequência de dois passos e σ_a^2 é a variância na aceleração durante o intervalo de um par de passos. A, B e C são parâmetros que tem que ser calibrados para a pessoa. Para cada combinação de posição do *smartphone* e forma de movimento, devem ser aprendidos os parâmetros A, B e C.

Estimação da direção

De acordo com (WANG et al., 2018) o *offset* do ângulo de direção entre a direção real do pedestre e a direção do *smartphone* pode ser obtido através de um método baseado em *Principal Component Analysis* (PCA), em que a direção é dada como o primeiro autovalor da direção do usuário.

3.6 *PROBABILISTIC CONTEXT-AWARE STEP LENGTH ESTIMATION FOR PEDESTRIAN DEAD RECKONING*

No trabalho desenvolvido por (MARTINELLI et al., 2018), é proposto um método para as etapas de detecção do passo e estimação do tamanho do passo para um sistema de PDR do tipo SHS, que utiliza classificação de modos de utilização para melhorar o desempenho geral do sistema.

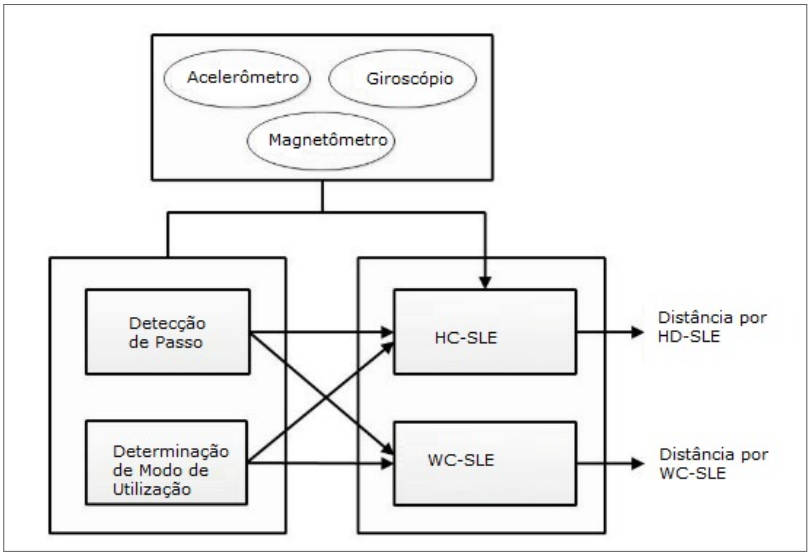
Assim como os demais trabalhos já listados aqui, ele também utiliza os sensores inerciais do *smartphone* (acelerômetro, giroscópio e magnetômetro). Para estimar o tamanho do passo duas técnicas são empregadas, *Weighted Context-Based Step Length Estimation* (WC-SLE) e *Highest Context-Based Step Length Estimation* (HC-SLE). Uma visão geral do sistema pode ser vista na Figura 27.

Classificação de modos de utilização

Para a classificação de modos de utilização, foram considerados os cenários descritos no Quadro 5. Em todos os casos, a posição do *smartphone* foi a mesma, que pode ser vista na Figura 28.

Para o processo de classificação, foram extraídas características a partir de uma janela de 2 segundos de duração com 50% de sobreposição. Foram geradas características do domínio do tempo, *range*, desvio padrão (σ), assimetria, curtose, energia, taxa *zero-crossing*, e domínio da frequência, pico máximo e índice do pico máximo.

Figura 27 – Visão geral do sistema proposto por (MARTINELLI et al., 2018).

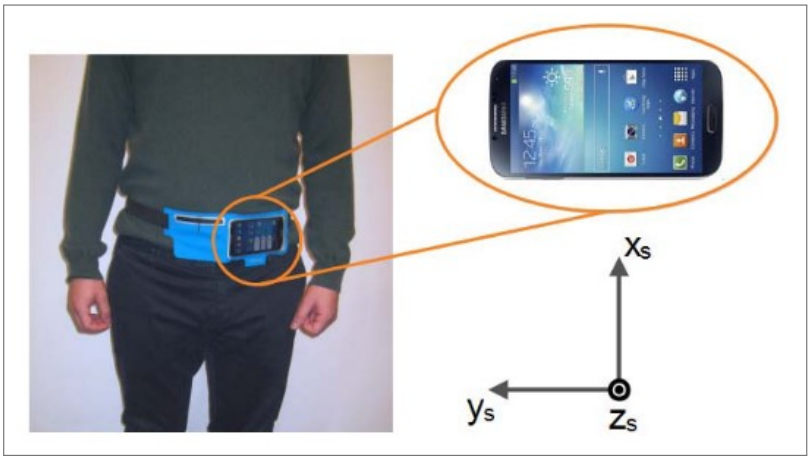


Fonte: Adaptado de (MARTINELLI et al., 2018).

Quadro 5 – Padrões de movimento considerados por (MARTINELLI et al., 2018).

Padrão de movimento	Descrição
C_1	Estacionário.
C_2	Andando.
C_3	Subindo escadas andando.
C_4	Descendo escadas andando.
C_5	Andando de lado.
C_6	Correndo.

Figura 28 – Posição do *smartphone* em todos os padrões de movimento em (MARTINELLI et al., 2018).



Fonte: (MARTINELLI et al., 2018).

Um resumo das características usadas pode ser vista no Quadro 6, em que a representa a leitura da magnitude do acelerômetro, a_f representa a leitura da magnitude do acelerômetro filtrado, g representa a leitura da magnitude do giroscópio e m representa a leitura da magnitude do magnetômetro.

Quadro 6 – Características geradas para classificação dos padrões de movimentos (MARTINELLI et al., 2018). Na coluna sinal, a representa a leitura da magnitude acelerômetro, a_f representa a leitura da magnitude do acelerômetro filtrado, g representa a leitura da magnitude do giroscópio e m representa a leitura da magnitude do magnetômetro.

Sinal	*Range*	σ	Assimetria	Curtose	Energia	Max Pico	Índice Max Pico	*zero-crossing*
a	x	x	x	x	x	x	x	x
a_f	x	x	x	x	x	x	x	x
g	x	x	x	x	x	x	x	
m	x	x	x	x	x			

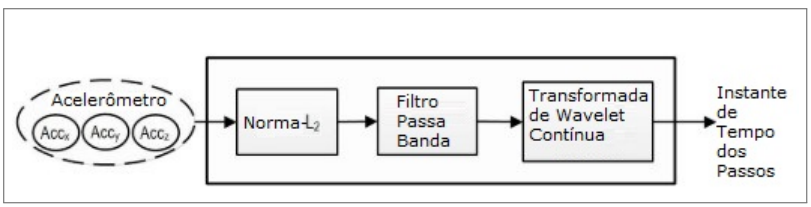
Para efetuar a classificação dentre os possíveis padrões de movimento, foi utilizada uma *Relevance Vector Machine* (RVM).

Detecção do passo

A detecção de passo é feita através das leituras de magnitude do acelerômetro. As leituras são armazenadas numa janela deslizante de duração de 2 segundos. Então, as leituras são filtradas por um filtro passa banda, para remover ruídos de alta frequência e para obter média zero.

Depois dessas etapas o sinal é enviado a um bloco que faz uma análise através de *Continuous Wavelet Transform* (CWT). Todo o processo pode ser visto na Figura 29. Através da análise CWT, é possível detectar os passos pela correlação entre o sinal de entrada e a *wavelet*.

Figura 29 – Diagrama de blocos da detecção de passo proposta por (MARTINELLI et al., 2018).



Fonte: Adaptado de (MARTINELLI et al., 2018).

Estimação do tamanho do passo

A estimação do passo é feita por uma combinação linear, definida como

$$SL = k_0 + k_1 f_s + k_2 \sigma_f^2 \tag{3.14}$$

em que, f_s é a frequência de passo, σ_f^2 é a variância da magnitude do acelerômetro. k_1, k_2 e k_0 são constantes que devem ser calibradas para cada um dos modos de utilização.

Para a estimação do tamanho são consideradas duas técnicas HC-SLE e WC-SLE. A distância percorrida é dada pela seguinte equação

$$d_{k,C_n} = \sum_{p=1}^{M} k_0^{C_n,p} + k_1^{C_n,p} f_{s,p} + k_2^{C_n,p} \sigma_{f,p}^2, \tag{3.15}$$

em que d_{k,C_n} indica a distância percorrida entre as épocas $k-1$ e k considerando o padrão de movimento C_n (Quadro 5). Uma época é definida como sendo a última estimação de padrão de movimento, uma vez que a classe é definida sobre uma janela deslizante com sobreposição de 50%. $k_0^{C_n,p}$, $k_1^{C_n,p}$ e $k_2^{C_n,p}$ são as constantes definidas para o padrão C_n e $f_{s,p}$ e $\sigma_{f,p}^2$ são a frequência de passo,e a variância da magnitude do acelerômetro no passo detectado p, M é o total de passos detectados dentro da época atual.

Para o método HC-SLE a distância percorrida é dada pelo d_{k,C_n} que teve o C_n mais provável. Já no método WC-SLE, a distância é dada pela média ponderada de cada d_{k,C_n} usando a probabilidade como peso.

3.7 COMPARAÇÃO DOS TRABALHOS

Em todos os trabalhos mencionados neste Capítulo, a menos do (HUANG et al., 2016), utilizaram algum mecanismo de aprendizagem de máquina para aumentar o desempenho do algoritmo domo um todo. Apenas o (MARTINELLI et al., 2018) teve posição fixa para o *smartphone*, o que mostra uma tendência de se considerar as várias possíveis posições que o *smartphone* pode tomar durante o processo de localização, seja em duas dimensões (2D) ou em três dimensões (3D).

Percebe-se o uso de todos os sensores inerciais do *smartphone* para o cálculo da localização seja, 2D ou 3D. No caso 3D, faz-se também necessário o uso do barômetro para que se possua uma referência absoluta de altitude. Comparar os trabalhos quantitativamente é uma tarefa muito complicada, uma vez que cada um possui uma base de dados própria não disponível, e ainda, as métricas são diferentes em alguns casos.

A falta de uma metodologia de avaliação padrão para os trabalhos de PDR dificulta muito a evolução dos trabalhos e a falta de bases de dados públicas, a sua

replicação. Também existe a falta de detalhamento de algumas partes críticas dos algoritmos, que é um agravante para a replicação, em alguns casos, apenas é dito que foi utilizado um filtro de Kalman.

Um comparativo dos trabalhos pode ser visto no Quadro 7. A coluna "Parâmetros que precisam ser calibrados"representa os parâmetros que precisam ser calibrados para cada pessoa. Alguns podem ser calibrados por meios de algoritmos, outros são manualmente.

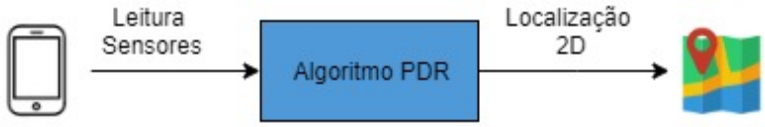
Quadro 7 – Quadro comparativo dos trabalhos vistos nas sessões anteriores. As abreviações acc, giro e mag indicam, respectivamente, acelerômetro, giroscópio e magnetômetro.

Ref.	Posição do Smatphone	Tipo	Sensores	Classificador	Fusão de Sensores	Parâmetros que precisam ser calibrados
(SHIN et al., 2012)	Fixa	PDR 3D	acc, giro, mag, barômetro	ANN		8
(PRATAMA; WIDYAWAN; HIDAYAT, 2012)	Fixa	Detecção de passo e estimação do tamanho do passo	acc			2
(MODER et al., 2015)	Fixa	PDR 2D	acc, giro, mag, barômetro, sinal de WiFi	Árvore de Decisão		4
(ELHOUSHI et al., 2016)	Variável	PDR 3D	acc, giro, mag, barômetro	Árvore de Decisão		
(SHIN et al., 2016)	Variável	PDR 3D	acc, giro, mag, barômetro, GPS	ANN	KF	6
(HUANG et al., 2016)	Variável	PDR 2D	acc, giro, mag		KF	
(KLEIN; SOLAZ; OHAYON, 2018)	Variável	PDR 2D	acc, giro	Gradient Boosting de Árvore de Decisão		1
(WANG et al., 2018)	Variável	PDR 2D	acc, giro, mag	SVM e Árvore de Decisão		48
(MARTINELLI et al., 2018)	Fixa	Detecção de passo e estimação do tamanho do passo	acc, giro, mag	RVM		18
(KUANG; NIU; CHEN, 2018)	Variável	PDR 2D	acc, giro, mag		EKF	

4 MÉTODO PROPOSTO

O sistema proposto neste trabalho tem como objetivo prover a localização (2D) do usuário para um possível sistema de chave passiva veicular *Passive Keyless Entry and Start* (PKES). Para prover a localização para o sistema PKES será utilizado um algoritmo de *Pedestrian Dead Reckoning* (PDR), que utilizará os sensores inerciais do *smartphone* (giroscópio, acelerômetro, magnetômetro e vetor de rotação), todos fornecidos pela *Application Programming Interface* (API) do sistema Android (Figura 30).

Figura 30 – Sistema PDR proposto.



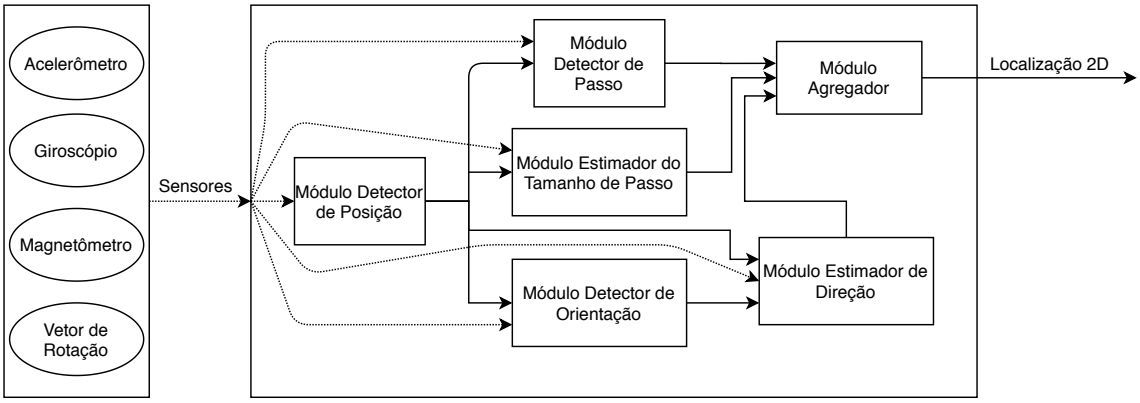
O sistema PKES então somente ligará o rádio (para de fato efetuar o protocolo de abertura do veículo) da chave uma vez que o sistema de PDR informe que a localização do usuário é próxima ao veículo. Desta forma, o sistema estará mais protegido contra ataques de *relay* (Seção 1.1), uma vez que não irá existir mais sinal de rádio para ser copiado e retransmitido para um atacante. O ataque só poderá ser realizado quando o usuário estiver perto do veículo, que neste caso o ataque em si não faria muito sentido, já que o objetivo do ataque é ser furtivo.

4.1 VISÃO GERAL DO ALGORITMO DE PDR PROPOSTO

O sistema de PDR proposto é um sistema do tipo SHS, e engloba as três etapas, detecção de passo, estimação do tamanho do passo e estimação da direção. A arquitetura geral do sistema pode ser vista na Figura 31.

As três etapas do sistema de SHS podem ser vistas nos módulos de detecção de passo, módulo estimador do tamanho de passo e módulo estimador de direção. Os módulos de detecção de posição e de detecção de orientação são módulos que servem para ajudar no desempenho dos módulos anteriores, dando informações da posição no corpo em que o *smartphone* está e a sua orientação.

Figura 31 – Visão Geral do Algoritmo de PDR Proposto.



4.2 GERAÇÃO DE CARACTERÍSTICAS

Nesta seção, serão descritas as técnicas utilizadas para a geração de características. Estas características são utilizadas nos demais módulos do algoritmo de PDR proposto, para os classificadores e regressores.

4.2.1 Janela Deslizante

Nos módulos de detecção de passo, detecção de posição, estimação de tamanho de passo e detecção de orientação, utiliza-se um conceito de janela deslizante que será apresentado nesta Seção.

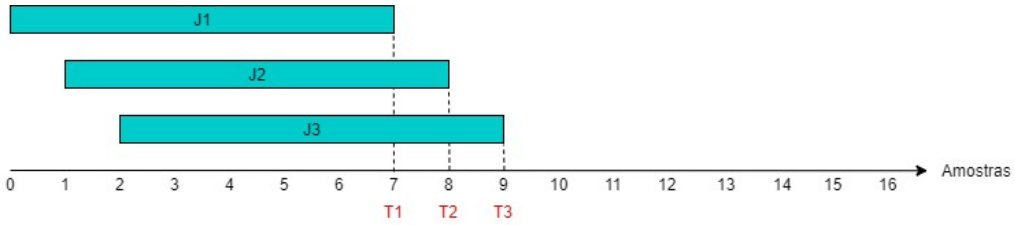
Uma janela deslizante é um tipo de janela que tem 2 parâmetros: 1) tamanho da janela e 2) sobreposição. O tamanho da janela é a quantidade de amostras que são armazenadas na janela, e a sobreposição é a quantidade de amostras novas que devem chegar nesta janela para que uma nova janela possa ser gerada.

Na Figura 32a, tem-se um exemplo de janela de tamanho 8 com 1 amostra de sobreposição. Neste caso, a cada nova amostra é gerada uma nova janela, os instantes que as janelas são geradas são representados por $T1$, $T2$ e $T3$. Já na Figura 32b temos uma janela de tamanho 8 com sobreposição de 4 amostras ou 50%. Com 50% de sobreposição as novas janelas levam mais tempo para serem formadas, como pode-se ver os instantes $T1$, $T2$ e $T3$ agora ocorrerem nos instantes 7, 11 e 15, diferente da janela da Figura 32a que eram em 7, 8 e 9.

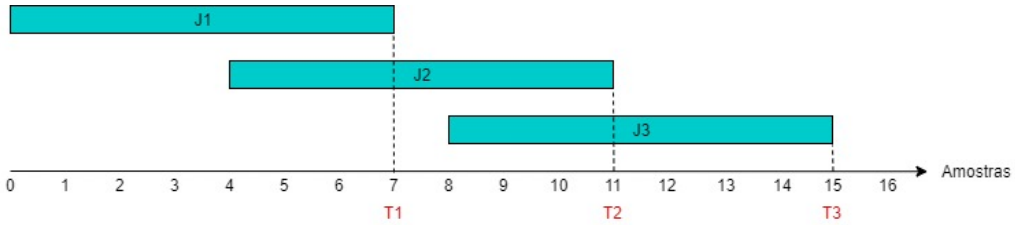
4.2.2 Características Geradas

Nos módulos de detecção de posição, estimação de tamanho de passo e detecção de orientação, são calculadas características dos sinais de entradas através de janelas deslizantes (Seção 4.2.1). Este processo gera características de duas naturezas

Figura 32 – Janela deslizante de tamanho 8 com duas configurações diferentes de sobreposição.



(a) Sobreposição de 1 amostra.



(b) Sobreposição de 4 amostras (50%).

diferentes, quais sejam, características do domínio do tempo e características do domínio da frequência.

Domínio do Tempo

As características do domínio do tempo utilizadas foram a média aritmética das leituras armazenadas na janela,

$$F_{mean} = \frac{1}{N} \sum_{i=0}^{N-1} W_i \tag{4.1}$$

em que W_i representa a amostra i na janela W, o desvio padrão,

$$F_{std} = \sqrt{\frac{1}{N} \sum_{i=0}^{N-1} \left(W_i - F_{mean}\right)^2} \tag{4.2}$$

o *range* dos valores dentro da janela W,

$$F_{range} = |\max W - \min W| \tag{4.3}$$

a curtose,

$$F_{kurt} = \frac{\frac{1}{N} \sum_{i=0}^{N-1} \left(W_i - F_{mean}\right)^4}{\left(\frac{1}{N} \sum_{i=0}^{N-1} \left(W_i - F_{mean}\right)^2\right)^2} - 3 \tag{4.4}$$

a assimetria,

$$F_{skew} = \frac{\frac{1}{N} \sum_{i=0}^{N-1} \left(W_i - F_{mean}\right)^3}{\left(\frac{1}{N} \sum_{i=0}^{N-1} \left(W_i - F_{mean}\right)^2\right)^{\frac{3}{2}}} \tag{4.5}$$

a energia,

$$F_{energy} = \sum_{i=0}^{N-1} W_i^2 \tag{4.6}$$

e a quantidade de vezes que houve uma troca de sinal dentro da janela

$$s(x,y) = \begin{cases} 1, & x \cdot y < 0 \\ 0, & cc \end{cases}$$
$$F_{ZeroCross} = \sum_{i=0}^{N-1} s\left(W_i, W_{i+1}\right) \tag{4.7}$$

Domínio da Frequência

As características do domínio da frequência utilizadas foram baseadas na *Fast Fourier Transform* (FFT) da janela. Foram criadas duas características, o maior valor da transformada

$$F_{FFT_{max}} = \max \mathscr{F}\{W\} \tag{4.8}$$

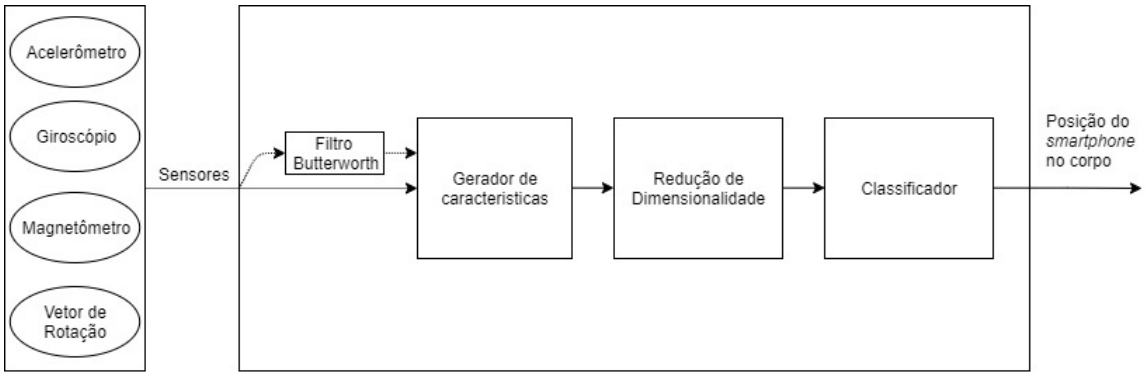
em que, $\mathscr{F}\{W\}$ representa a FFT da janela, e o índice de maior valor

$$F_{FFT_{idx}} = argmax\, \mathscr{F}\{W\} \tag{4.9}$$

4.3 DETECÇÃO DE POSIÇÃO DO *SMARTPHONE*

O módulo detector de posição tem como função definir em que posição no corpo do usuário está o *smartphone*. A resposta deste módulo é fundamental para aumentar a acurácia do sistema como um todo, uma vez que permite a seleção de parâmetros mais apropriados para as demais etapas. Uma visão geral do módulo detector de posição pode ser vista na Figura 33.

Figura 33 – Módulo detector de posição. Responsável por detectar a posição do *smartphone* no corpo do usuário.



Filtro Butterworth

Este bloco tem como finalidade remover ruídos de alta frequência no sinal do acelerômetro. Para isto, é empregado um filtro de Butterworth passa baixa de ordem

5 com frequência de corte em 5Hz. O sinal filtrado também é utilizado na etapa de detecção de passo.

Gerador de Características

O bloco gerador de características tem como entrada, os sinais vindos do acelerômetro, do giroscópio e do magnetômetro. Destes sinais, são utilizados as seguintes componentes: magnitude do acelerômetro (a), magnitude do acelerômetro filtrada (a_f), magnitude do giroscópio (g), magnitude do magnetômetro (m), componente x no *frame* da terra do acelerômetro (a_x^E), componente x no *frame* da Terra do giroscópio (g_x^E), componente y no *frame* da Terra do giroscópio (g_y^E) e componente z no *frame* da Terra do giroscópio (g_z^E). As características geradas por este módulo estão sumarizadas no Quadro 8.

Quadro 8 – Características utilizadas pelo classificador de posição do *smartphone*.

Sinal	F_{mean}	F_{std}	F_{range}	F_{kurt}	F_{skew}	F_{energy}	$F_{FFT_{idx}}$	$F_{FFT_{max}}$	$F_{ZeroCross}$
a	X	X	X	X	X	X	X	X	X
a_f	X	X	X	X	X	X	X	X	X
g	X	X	X	X	X	X	X	X	
m	X	X	X	X	X	X			
a_x^E	X	X	X	X	X	X			
g_x^E	X	X	X	X	X	X			
g_y^E	X	X	X	X	X	X			
g_z^E	X	X	X	X	X	X			

Redução de Dimensionalidade

Antes dos dados criados pelo gerador de características serem entregues para o classificador, os mesmos passam por um processo de redução de dimensionalidade para serem executados de maneira mais eficiente pelo classificador. A técnica utilizada para fazer esta redução de dimensionalidade foi *Principal Component Analysis* (PCA).

Classificador

O classificador é responsável por determinar em que local o *smartphone* está localizado no corpo do usuário dentro de 5 possíveis posições. As posições consideradas neste trabalho foram: *Pocket*, *BackPocket*, *Hand*, *Texting* e *Head*. A descrição detalhada de cada uma das possíveis posições pode ser vista no Quadro 9.

Quadro 9 – Posições onde o *smartphone* pode se encontrar no corpo do usuário.

Posição	Descrição
Pocket	O *smartphone* está localizado no bolso direito da frente da calça.
BackPocket	O *smartphone* está localizado no bolso de trás direito da frente da calça.
Hand	O *smartphone* está localizado na mão direita do usuário.
Texting	O *smartphone* está localizado na mão direita do usuário em uma posição de utilização.
Head	O *smartphone* está localizado na mão direita colada a cabeça em posição utilizada durante uma ligação.

4.4 DETECÇÃO DE PASSO

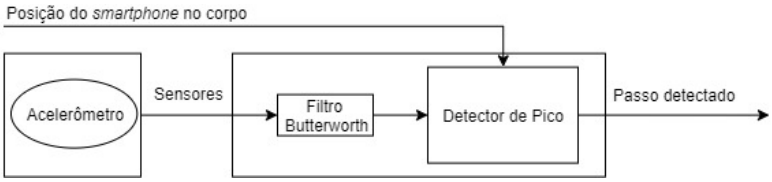
Assim como no módulo detector de posição, uma janela deslizante é utilizada. A janela deslizante armazena as leituras da magnitude do acelerômetro filtradas (mesmo filtro utilizado em Seção 4.3). Estas leituras serão utilizadas por um algoritmo de detecção de pico.

O algoritmo de detecção de pico, além da janela, necessita de um limiar (λ). Este limiar é definido a partir da posição em que o *smartphone* foi detectado. De posse da janela e do limiar, o Algoritmo 1 é executado para fazer a detecção de passo. Uma visão geral do módulo detector de passo pode ser vista na Figura 34.

Algoritmo 1: Detector de Pico

1: $Detector\,de\,Pico\,(janela:\,W,\,limiar:\,\lambda)$
2: $centro \leftarrow W\left[\frac{len\,W-1}{2}\right]$
3: **if** $(centro = \max W) \wedge (centro > \lambda) \wedge (centro > mean\,W)$ **then**
4: Passo Detectado
5: **end if**

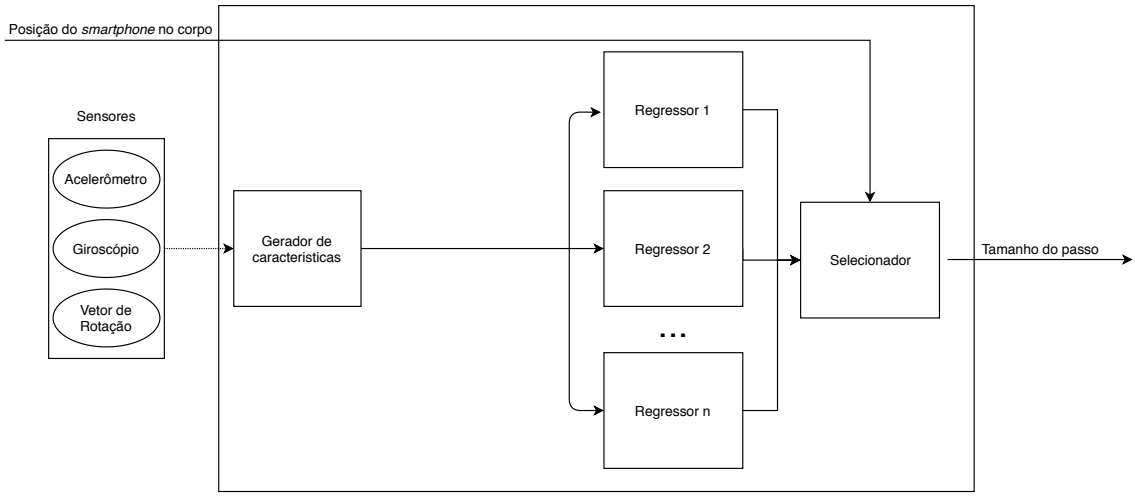
Figura 34 – Módulo detector de passo, responsável por detectar os instantes em que ocorre um passo do usuário.



4.5 ESTIMAÇÃO DO TAMANHO DO PASSO

O módulo estimador de passo tem como função estimar em metros a distância percorrida por cada passo do usuário. Para aumentar sua acurácia, foram treinados vários regressores, cada um especializado em um posição do *smartphone*. Para se definir a posição do *smartphone*, utiliza-se a informação de posição calculada no módulo detector de posição do *smartphone* (Seção 4.3). Uma visão geral do módulo de estimação do tamanho do passo pode ser vista na Figura 35.

Figura 35 – Módulo estimador de tamanho de passo.



Gerador de Características

O bloco gerador de características tem como entrada os sinais do acelerômetro e do giroscópio. Destes sinais, são utilizadas as seguintes componentes: magnitude do acelerômetro (a), magnitude do giroscópio (g), componente x no *frame* da terra do acelerômetro (a_x^E), componente x no *frame* da Terra do giroscópio (g_x^E), componente y no *frame* da Terra do giroscópio (g_y^E) e componente z no *frame* da Terra do giroscópio (g_z^E). As características geradas por este módulo são sumarizadas no Quadro 10.

Regressor

Os regressores são os módulos que de fato vão fazer a estimação da distância percorrida. Para cada uma das possíveis posições do *smartphone* existe um regressor especializado. Todos os regressores possuem a mesma entrada e são treinados apenas usando dados referente à posição que são especializados.

Quadro 10 – Características utilizadas pelo regressor de tamanho de passo.

Sinal	F_{mean}	F_{std}	F_{range}	F_{kurt}	F_{skew}	F_{energy}
a	X	X	X	X	X	X
g	X	X	X	X	X	X
a_x^E	X	X	X	X	X	X
g_x^E	X	X	X	X	X	X
g_y^E	X	X	X	X	X	X
g_z^E	X	X	X	X	X	X

Selecionador

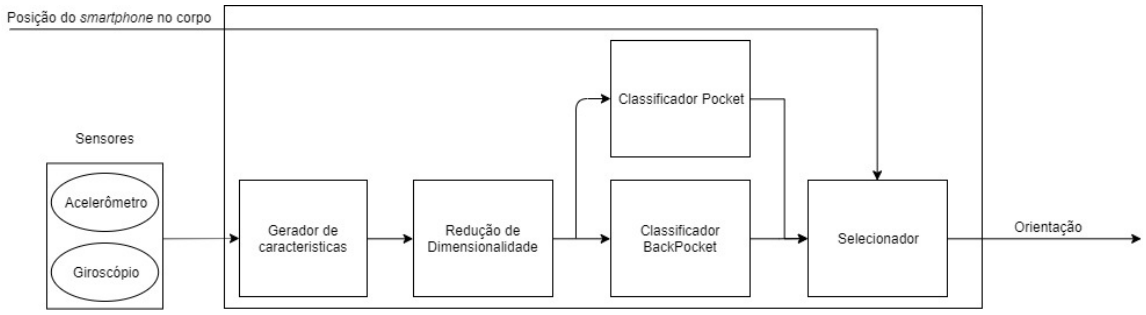
O módulo selecionador nada mais é que um seletor que, dada a posição do *smartphone* no corpo do usuário, irá selecionar como saída a resposta do regressor especializado para aquela posição.

4.6 DETECÇÃO DE ORIENTAÇÃO DO *SMARTPHONE*

Esta etapa tem como objetivo determinar a orientação do *smartphone*. Esta informação é necessária para a etapa de estimação de direção. São consideradas quatro possíveis orientações para as posições *Pocket* e *BackPocket*: P1 (Figura 37a), P2, (Figura 37b), P3(Figura 37c) e P4(Figura 37d). Para as orientações P1 e P2, a tela está virada para fora do bolso, e para as posições P3 e P4, para dentro do bolso, seja o bolso da frente ou o de trás.

O módulo responsável por esta etapa é o modulo detector de orientação (Figura 31). Uma visão geral dele pode ser vista na Figura 36.

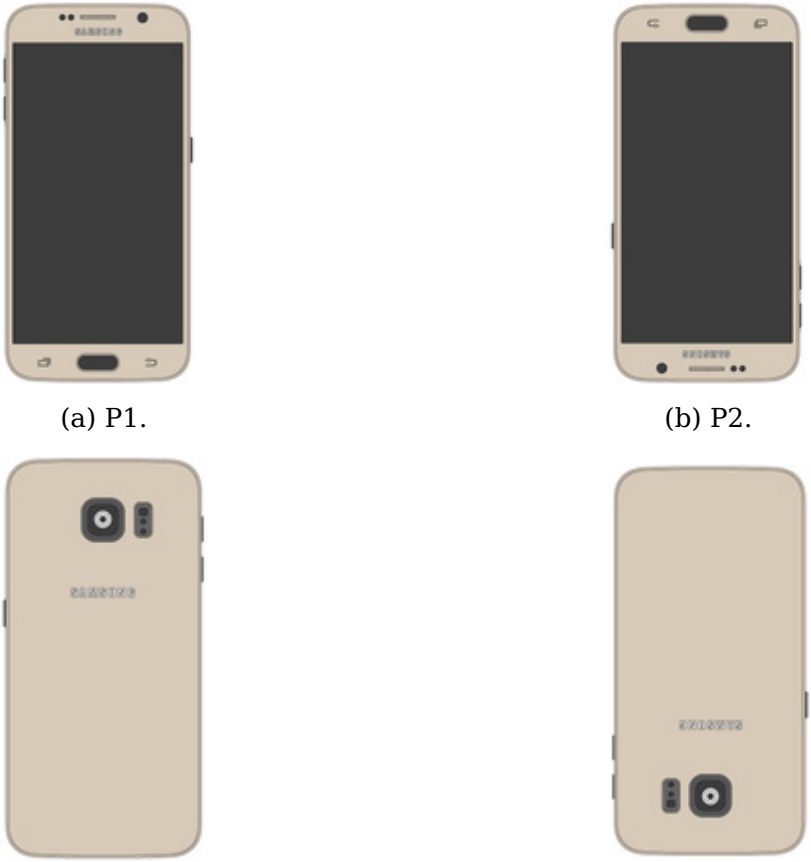
Figura 36 – Módulo detector de orientação do *smartphone*.



Gerador de Características

O bloco gerador de características tem como entrada os sinais do acelerômetro e giroscópio. Destes sinais, são utilizados as seguintes componentes: magnitude

Figura 37 – Orientações do *smartphone*.



(a) P1. (b) P2.

(c) P3. (d) P4.

do acelerômetro (a), componente x do acelerômetro (a_x), componente y do acelerômetro (a_y), componente z do acelerômetro (a_z), magnitude do giroscópio (g), componente x do giroscópio (g_x), componente y do giroscópio (g_y) e componente z do giroscópio (g_z). As características geradas por este módulo são sumarizadas no Quadro 11.

Quadro 11 – Características utilizadas pelo classificador de orientação do *smartphone*.

Sinal	F_{mean}	F_{std}	F_{range}	F_{kurt}	F_{skew}	F_{energy}	$F_{FFT_{idx}}$	$F_{FFT_{max}}$
a	X	X	X	X	X	X	X	X
a_x	X	X	X	X	X	X	X	X
a_y	X	X	X	X	X	X	X	X
a_z	X	X	X	X	X	X	X	X
g	X	X	X	X	X	X	X	X
g_x	X	X	X	X	X	X	X	X
g_y	X	X	X	X	X	X	X	X
g_z	X	X	X	X	X	X	X	X

Redução de Dimensionalidade

Assim como no módulo de detecção de posição do *smartphone*, é empregada uma redução de dimensionalidade. A técnica utilizada também é o PCA, com o mesmo intuito, de fazer com que as características geradas possam ser processadas de maneira mais eficiente pelos módulos seguintes, o classificador de orientação *Pocket* e *BackPocket*.

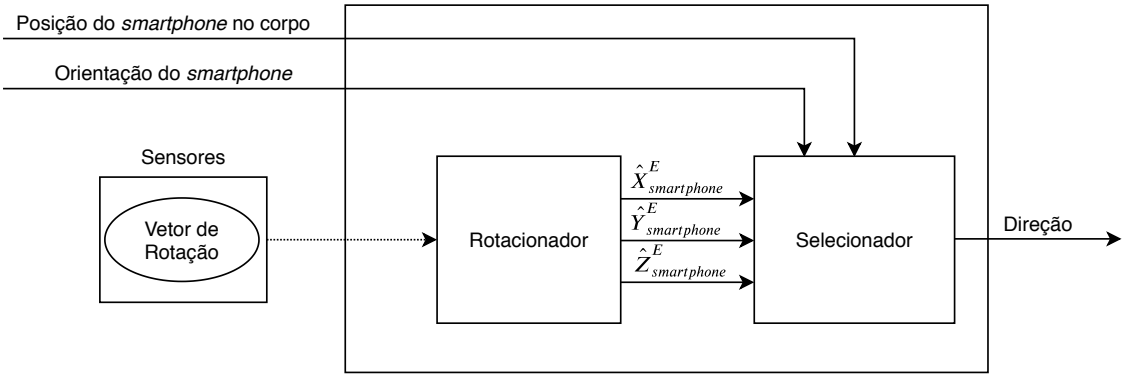
Selecionador

Este bloco tem como objetivo selecionar qual dos classificadores será utilizado, para isto utiliza-se da informação de posição do *smartphone*. Nos casos em que a posição não for *Pocket* ou *BackPocket*, a **orientação retornada é sempre a mesma**, P1.

4.7 ESTIMAÇÃO DE DIREÇÃO

A estimação de direção é feita pelo módulo estimador de direção. Para a estimação da direção são consideradas as seguintes informações: a posição do *smartphone*, a orientação do *smartphone* e o vetor de rotação fornecido pela API do Android. Uma visão geral do módulo pode ser vista na Figura 38.

Figura 38 – Módulo estimador de direção do usuário.

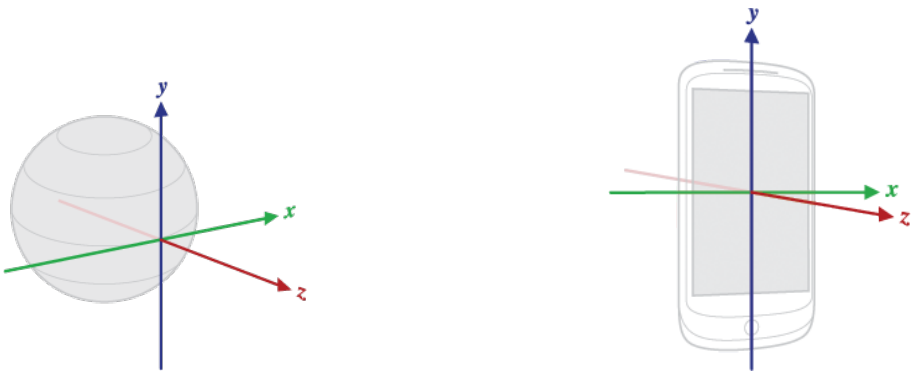


Rotacionador

O rotacionador tem como função representar os eixos do *smartphone* no *frame* de referência do globo. Para isto, utiliza-se a informação do vetor de rotação que armazena a rotação que leva os eixos do *smartphone* para o *frame* de referência do globo. Para efetuar a rotação dos vetores, utiliza-se as operações com quaterniões (Seção 2.3.2).

Os eixos considerados para o globo e para o *smartphone* pela API do Android podem ser vistos na Figuras 39a e 39b, respectivamente. Como saída do módulo tem-se $\hat{X}^{E}_{smartphone}$, $\hat{Y}^{E}_{smartphone}$ e $\hat{Z}^{E}_{smartphone}$ que são, respectivamente, os eixos x, y e z do *smartphone* no *frame* do globo.

Figura 39 – Eixos de referência fornecidos pela API do Android.



(a) Eixos de referência do globo. (b) Eixos de referência do *smartphone*.

Fonte: API do Android.

Selecionador

O selecionador tem como função definir qual é a direção de movimento do usuário em relação ao globo, dados a posição do *smartphone*, a orientação do *smartphone* e os vetores de saída do rotacionador. Para isto, é executado o Algoritmo 2.

4.8 MÓDULO AGREGADOR

O módulo agregador é o último módulo e tem como função agregar os resultados das três etapas do sistema SHS. Para isto, ele fica monitorando o sinal de detecção de passo fornecido pelo módulo detector de passo. Uma vez que um passo é detectado, as informações atuais de direção (Seção 4.7) e tamanho do passo (Seção 4.5) são utilizadas para fazer a atualização da posição do usuário.

Algoritmo 2: Selecionador de vetor de direção

1: $Definir\ direção\left(posição,\ orientação,\ \hat{X}^{E}_{smartphone},\ \hat{Y}^{E}_{smartphone},\ \hat{Z}^{E}_{smartphone}\right)$
2: **if** $posição\ smartphone\ =\ Head$ **then**
3: $\quad VecDireção \leftarrow -\hat{X}^{E}_{smartphone}$
4: **else if** $posição\ smartphone\ =\ Hand$ **then**
5: $\quad VecDireção \leftarrow \hat{Y}^{E}_{smartphone}$
6: **else if** $posição\ smartphone\ =\ Pocket$ **then**
7: $\quad$ **if** $orientação\ smartphone\ =\ P1$ **then**
8: $\quad\quad VecDireção \leftarrow \hat{Z}^{E}_{smartphone}$
9: $\quad$ **else if** $orientação\ smartphone\ =\ P2$ **then**
10: $\quad\quad VecDireção \leftarrow \hat{Z}^{E}_{smartphone}$
11: $\quad$ **else if** $orientação\ smartphone\ =\ P3$ **then**
12: $\quad\quad VecDireção \leftarrow -\hat{Z}^{E}_{smartphone}$
13: $\quad$ **else if** $orientação\ smartphone\ =\ P4$ **then**
14: $\quad\quad VecDireção \leftarrow -\hat{Z}^{E}_{smartphone}$
15: $\quad$ **end if**
16: **else if** $posição\ smartphone\ =\ BackPocket$ **then**
17: $\quad$ **if** $orientação\ smartphone\ =\ P1$ **then**
18: $\quad\quad VecDireção \leftarrow -\hat{Z}^{E}_{smartphone}$
19: $\quad$ **else if** $orientação\ smartphone\ =\ P2$ **then**
20: $\quad\quad VecDireção \leftarrow -\hat{Z}^{E}_{smartphone}$
21: $\quad$ **else if** $orientação\ smartphone\ =\ P3$ **then**
22: $\quad\quad VecDireção \leftarrow \hat{Z}^{E}_{smartphone}$
23: $\quad$ **else if** $orientação\ smartphone\ =\ P4$ **then**
24: $\quad\quad VecDireção \leftarrow \hat{Z}^{E}_{smartphone}$
25: $\quad$ **end if**
26: **else if** $posição\ smartphone\ =\ Texting$ **then**
27: $\quad VecDireção \leftarrow \hat{Y}^{E}_{smartphone}$
28: **end if**
29: $retorne\ VecDireção$

5 *SETUP* EXPERIMENTAL

Neste Capítulo, será detalhada a base de dados construída para os experimentos realizados com o método proposto, que foi detalhado no Capítulo 4.

Esta base de dados foi construída utilizando 2 modelos de *smarpthones*, Samsung Galaxy S6 e Zenfone 3 5.2". Para efetuar a coleta dos dados no *smarpthone*, foi construído um aplicativo Android.

Duas bases de dados foram geradas, uma controlada, que foi utilizada para treinamento dos classificadores e regressores, e uma não controlada para avaliar a generalização dos modelos em ambientes não controlados. Foi considerado apenas um usuário para a geração das bases de dados, tanto na controlada quanto na não controlada.

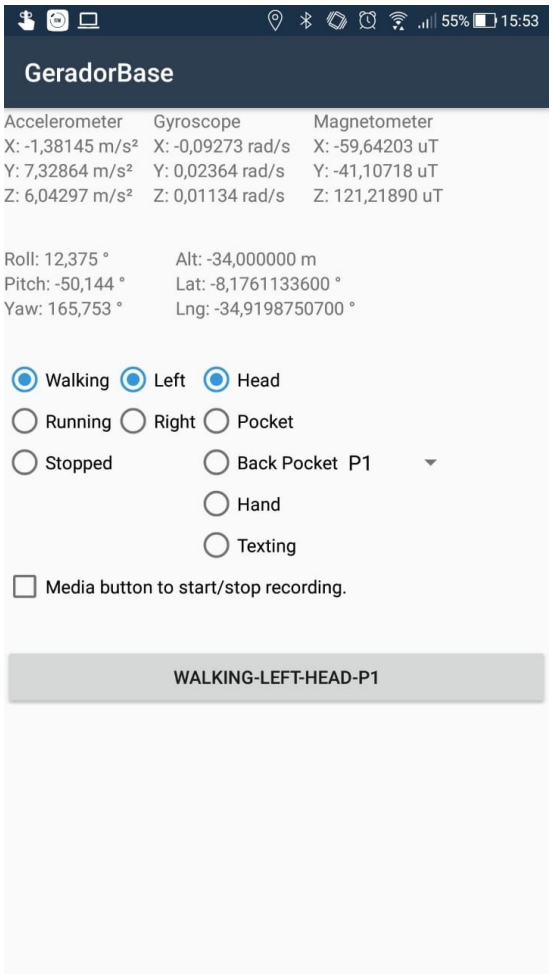
5.1 APLICATIVO ANDROID

O aplicativo para o sistema Android foi escrito em C# utilizando a plataforma Xamarin. O nível mínimo de *Application Programming Interface* (API) Android exigida para executar o aplicativo é a 21 (Lollipop 5.0). Uma visão da interface do aplicativo pode ser vista na Figura 40.

O usuário tem a opção de escolher qual a posição e orientação que o *smartphone* assumirá durante o experimento. Após a definição da posição, existem duas opções para iniciar o experimento: (1) através do botão principal da interface (botão com a posição do *smartphone*), (2) ou através do botão de mídia (botão que fica localizado nos fones de ouvido).

Os sinais logados pelo aplicativo estão definidos no Quadro 12. Todos os sinais são logados a uma taxa de 100Hz (10ms). O formato em que são armazenados para serem processados futuramente é o JSON. Um exemplo de amostra logada pode ser vista na Figura 41. Para se evitar arquivos muito grandes (maiores que 12MB), as *trips* (o trajeto percorrido), são quebradas em arquivos JSON de 5 minutos de duração.

Figura 40 – Tela do aplicativo Android para coletar a base de dados.



Quadro 12 – Sinais logados pelo aplicativo Android.

Sinal	Descrição
AccX	Componente X do acelerômetro em m/s^2.
AccY	Componente Y do acelerômetro em m/s^2.
AccZ	Componente Z do acelerômetro em m/s^2.
GyroX	Componente X do giroscópio em rad/s.
GyroY	Componente Y do giroscópio em rad/s.
GyroZ	Componente Z do giroscópio em rad/s.
MagX	Componente X do magnetômetro em μt.
MagY	Componente Y do magnetômetro em μt.
MagZ	Componente Z do magnetômetro em μt.
Roll	Ângulo Roll em radianos, $-\pi/2$ a $\pi/2$.
Pitch	Ângulo Pitch em radianos, $-\pi$ a π.
Yaw	Ângulo Yaw (Azimuth) em radianos, $-\pi$ a π.
Altitude	Valor da altitude em metros fornecida pelo GPS.
Latitude	Valor da latitude em graus fornecida pelo GPS.
Longitude	Valor da longitude em graus fornecida pelo GPS.
Q0	Componente 0 do vetor de rotação, adimensional.
Q1	Componente 1 do vetor de rotação, adimensional.
Q2	Componente 2 do vetor de rotação, adimensional.
Q3	Componente 3 do vetor de rotação, adimensional.

Figura 41 – Amostra coletada pelo pelo aplicativo no formato JSON.

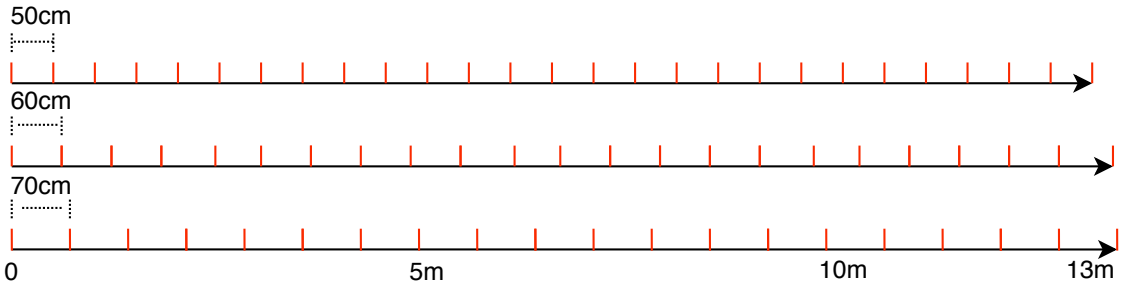

```
{
    "SamplingTime": "2019-11-10T19:48:25.519266-03:00",
    "AccX": 0.258575439,
    "AccY": -9.586395,
    "AccZ": 0.4477234,
    "GyroX": -0.0009460449,
    "GyroY": 0.0724029541,
    "GyroZ": 0.008056641,
    "MagX": -1.01470947,
    "MagY": -8.816528,
    "MagZ": 19.732666,
    "Roll": -0.6454788,
    "Pitch": 1.484888,
    "Yaw": 0.7109479,
    "Altitude": 7.0,
    "Latitude": -8.17607992,
    "Longitude": -34.9200797,
    "Q0": 0.729720354,
    "Q1": -0.6824318,
    "Q2": 0.00407091156,
    "Q3": -0.04217179
}
```

5.2 BASE CONTROLADA

Na base controlada foram feitos três trajetos curtos em linha reta: um de 13m, com 26 passos de 50cm; um de 13.2m, com 22 passos de 60cm e um de 13.3m, com 19 passos de 70cm (Figura 42).

Esta base serve para treinar os classificadores de detecção de posição e de orientação tanto quantos os regressores de tamanho de passo.

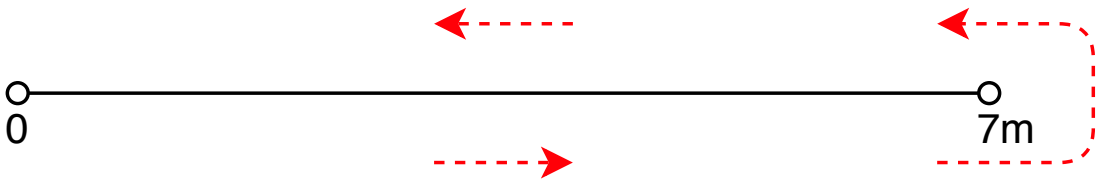
Figura 42 – Trajeto controlado em linha reta para três tamanhos de passos, 50cm, 60cm e 70cm.



5.3 BASE NÃO CONTROLADA CURTA

Esta é a primeira base para avaliar o desempenho do sistema em trajetos curtos ($14m$ de comprimento). O trajeto é uma linha reta com ida e volta para a origem. O trajeto pode ser visto na Figura 43.

Figura 43 – Trajeto curto não controlado de 14m.



5.4 BASE NÃO CONTROLADA LONGA

Esta base serve para avaliar o desempenho do sistema em trajetos longos na ordem de quilômetros. Foram definidos dois trajetos para esta base. O primeiro que varia de 3 a 3.8Km (Figura 44), em linha reta com volta à origem, que foi utilizado para avaliar o desempenho quando o *smartphone* está nas posições *Pocket* e *BackPocket*. E o segundo que varia de 0.9 a 1.2Km, com ida e volta à origem três vezes, que serviu para avaliar o desempenho quando o *smartphone* esta nas posições *Hand*, *Texting* e *Head*.

Figura 44 – Trajeto para avaliar o desempenho do sistema nas posições *Pocket* e *BackPocket*.

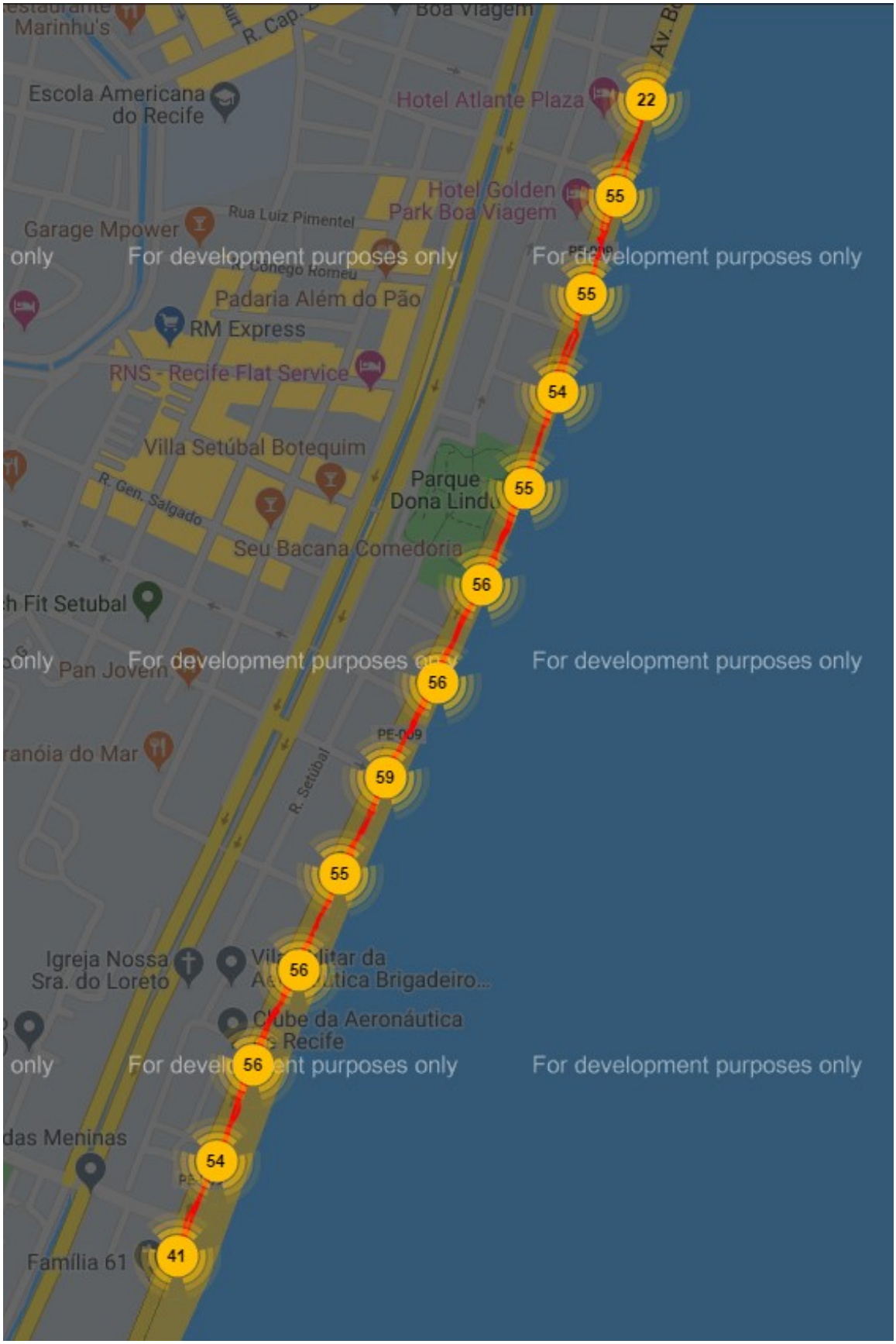
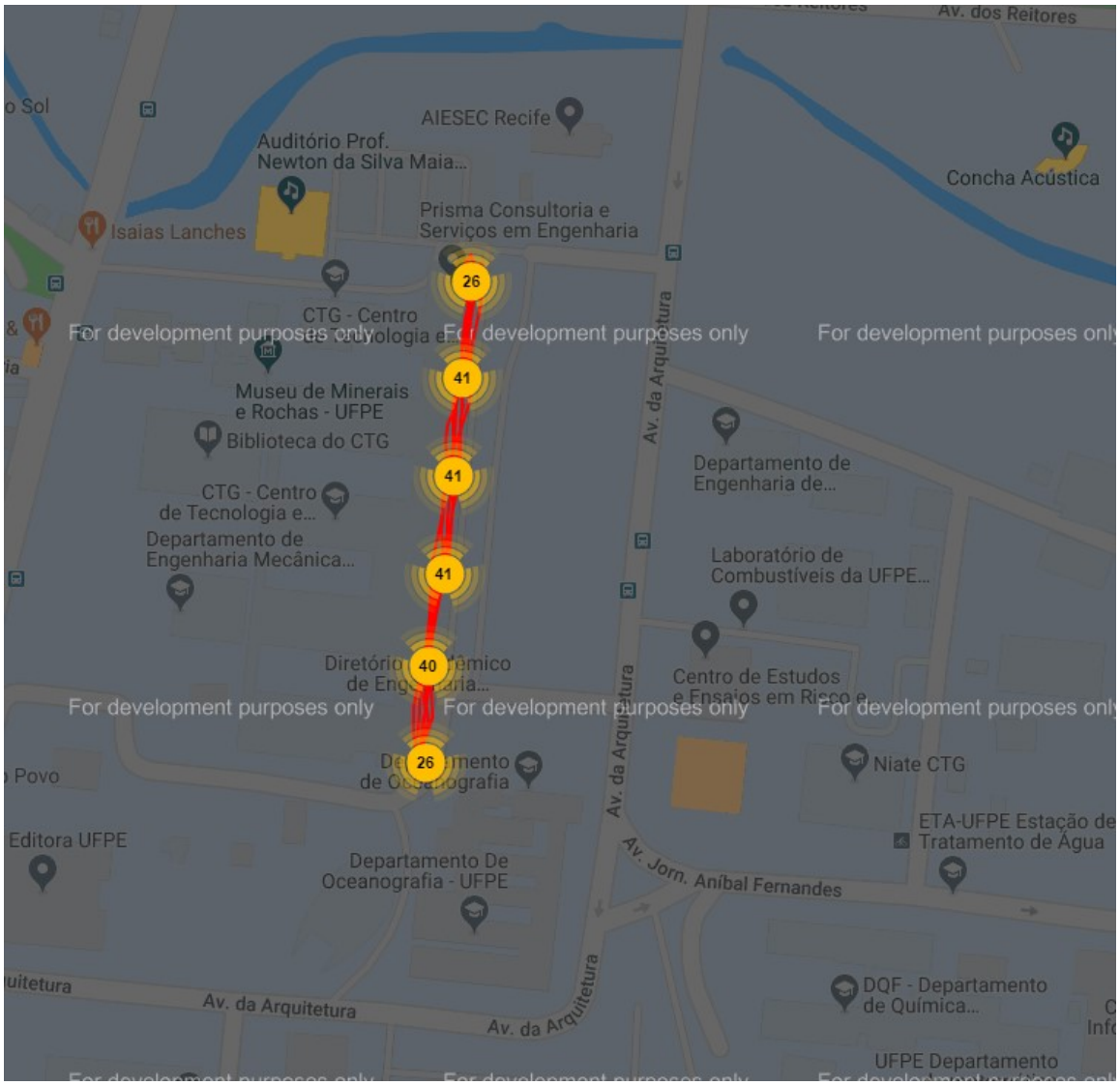


Figura 45 – Trajeto para avaliar o desempenho do sistema nas posições *Hand, Texting* e *Head*.



6 RESULTADOS E DISCUSSÕES

Este Capítulo é dedicado a apresentar os resultados obtidos, aplicando-se diferentes técnicas de classificação e regressão, no algoritmo de *Pedestrian Dead Reckoning* (PDR) proposto no Capítulo 4. Para isto, utilizou-se a base de dados descrita no Capítulo 5.

Nas Seções 6.1 e 6.2, são descritas as métricas e a metodologia de treino utilizada. Nas Seções 6.4 e 6.6, são avaliados diversos tipos de classificadores para as etapas de classificação de posição e orientação do *smartphone*. Na Seção 6.5, são avaliados diferentes tipos de regressores utilizados para definição do tamanho do passo. E por fim, na Seção 6.7, tem-se o resultado final do algoritmo de PDR proposto, integrando os melhores regressores e classificadores encontrados nas Seções anteriores.

6.1 MÉTRICAS UTILIZADAS

Nesta Seção, serão detalhadas as métricas utilizadas para avaliação dos modelos testados no algoritmo de PDR proposto.

Matriz de Confusão

A matriz de confusão é uma tabela especial que é utilizada para avaliar o desempenho de um classificador. Em cada linha, tem-se as classes preditas para uma determinada classe. Na Figura 46, tem-se um matriz de confusão para um população que possui três classes. Por exemplo, na primeira linha e primeira coluna, tem-se quantos padrões da classe A foram classificados como A. Na primeira linha e segunda coluna, tem-se quantos padrões da classe A foram classificados como B e na primeira e linha terceira coluna, tem-se quantos padrões da classe A foram classificados como C. Nas demais aplica-se o mesmo princípio.

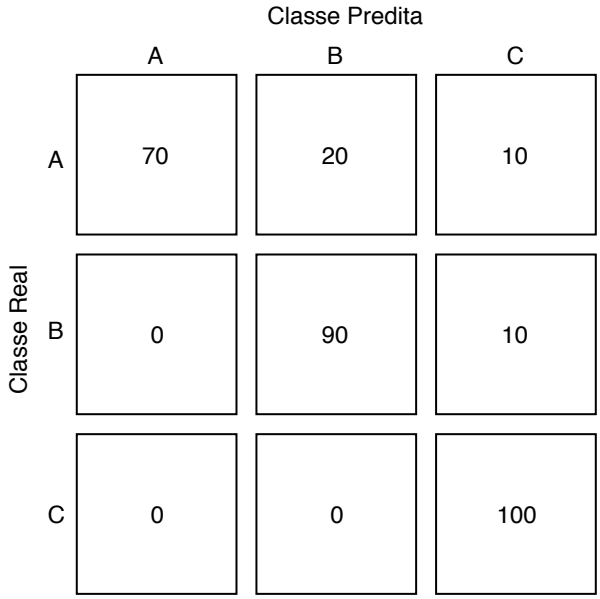
Acurácia

O desempenho de um classificador pode ser avaliada através da sua acurácia. A acurácia é definida como,

$$ACC(\hat{Y}) = \frac{\sum Padrões\, Classificados\, Corretamente}{\sum População\, Total} \tag{6.1}$$

em que $\hat{Y}$ são as predições feita por um classificador sobre toda a população. A acurácia pode ser entendida como sendo a fração de padrões que foram classificados corretamente dentro de uma população. Considerando a matriz de confusão, a

Figura 46 – Matriz de confusão para 3 classes.



acurácia seria a soma dos valores da diagonal principal divida pela soma de todos os valores da matriz.

RMSE

O *Root-Mean-Square Error* (RMSE) é uma métrica utilizada para se avaliar o desempenho de um regressor. É definido como

$$RMSE(\hat{Y}) = \sqrt{\frac{\sum_{t=1}^{T} \left(y_t - \hat{y}_t\right)^2}{T}} \tag{6.2}$$

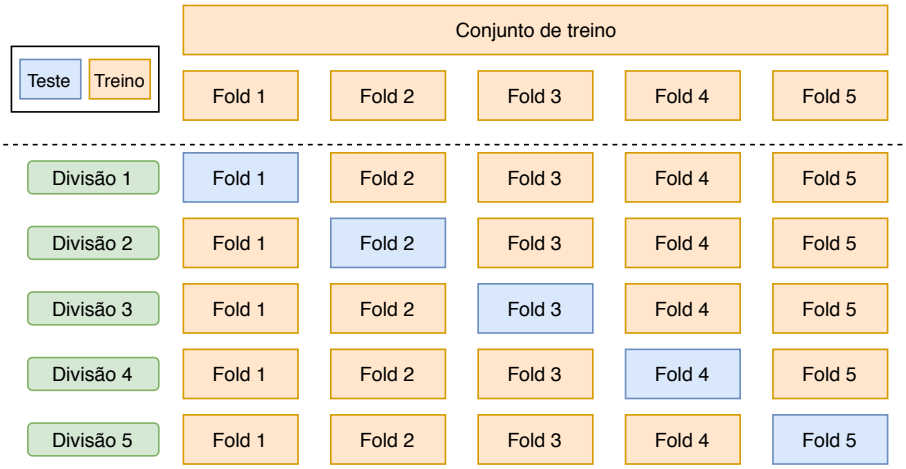
em que y_t representa o valor real, $\hat{y}_t$ o valor predito, e $\hat{Y}$, o conjunto de valores preditos. Por ser uma métrica dependente da escala dos dados, não pode ser utilizada para comparar regressores sobre diferentes tipos de dados. Por ser uma métrica proporcional ao quadrado do erro, é bastante sensível a *outliers*.

6.2 METODOLOGIA DE TREINO

Para o treinamento dos estimadores (classificadores e regressores), foi utilizada a técnica de validação cruzada K-Fold. A técnica consiste em embaralhar o conjunto de treino e dividir em K pedaços (Folds), gerando K testes, procedimento ilustrado na Figura 47.

Cada um dos pedaços é utilizado 1 vez como conjunto de teste, e $(K-1)$ vezes como conjunto treinamento. Desta forma, todos os pedaços são utilizados tanto para treino como para teste. Este procedimento serve para definir os hiperparâmetros dos estimadores.

Figura 47 – Validação cruzada do tipo K-Fold com $K = 5$.



6.3 DETECÇÃO DE PASSO

Como descrito na Seção 4.4 o algoritmo de detecção de passo precisa de um limiar (λ) que varia de acordo com a posição. Para definir o limiar e o tamanho da janela para detecção de passo, foi utilizada a base controlada (Seção 5.2).

Para definir o limiar e o tamanho da janela foi avaliado todo o espaço de possibilidades dentro do conjunto $S = J \times T$, em que o conjunto,

$$T = \{10.5; 10.59; 10.69; 10.78; ...; 20\}$$

(10.5 até 20 dividido em 100 partes) representa os valores de limiar, e

$$J = \{13, 15, 17, 19, 21, 23\}$$

representa os valores de tamanho de janela. Na Figura 48, tem-se a função de erro gerada para *BackPocket* com orientação P2, a função de erro é definida como,

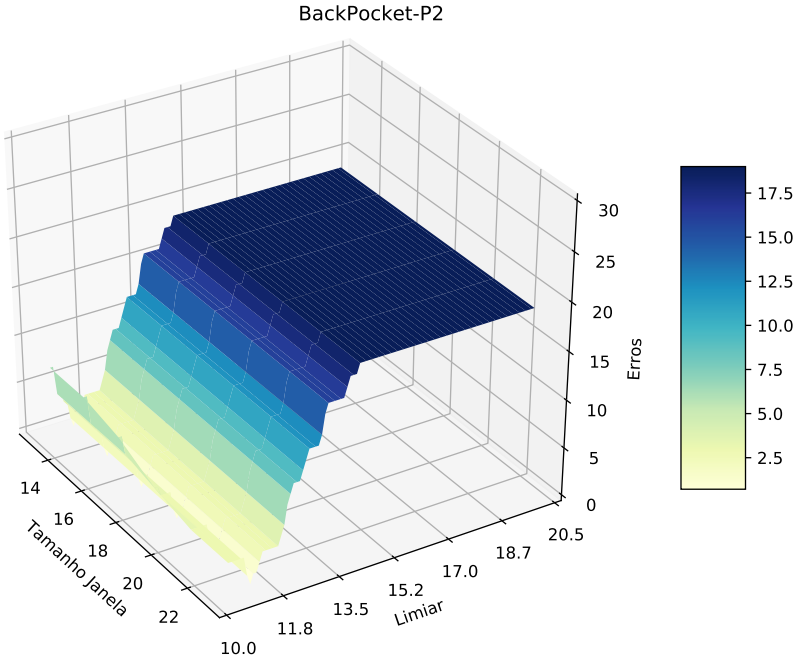
$$Erro = |Quantidade\,passos\,detectados - Quantidade\,passos\,real|$$

. Percebe-se que o tamanho da janela tem menos importância quando comparado com o valor do limiar.

Depois de calcular todo o espaço de possibilidades para todas as combinações de posição de *smartphone*, orientação de *smartphone* e tamanho de passo da base controlada, foi calculada a quantidade de erros gerados por cada tamanho de janela (utilizando o melhor valor de limiar correspondente). O resultado deste cálculo está na Tabela 3. Analisando a tabela, foi escolhido o tamanho de janela 17 como sendo a melhor opção por ter menos erros e ser menor do que as demais com mesma quantidade de erros.

Analisando mais a fundo a janela de tamanho 17, verificou-se na Tabela 4 os melhores valores de limiar de cada combinação de posição e orientação de *smartphone*.

Figura 48 – Superfície de detecção de passo para a posição *BackPocket* com orientação P2.



Tamanho do Passo	Tamanho da Janela	Quantidade de Erros
70cm	13	3
70cm	15	3
70cm	17	2
70cm	19	2
70cm	21	2
70cm	23	2
60cm	13	3
60cm	15	3
60cm	17	3
60cm	19	3
60cm	21	3
60cm	23	3
50cm	13	5
50cm	15	5
50cm	17	5
50cm	19	5
50cm	21	5
50cm	23	5

Tabela 3 – Quantidade de erros para cada tamanho de janela.

Posição no Corpo	Posição do Smartphone	Limiar em m/s^2		
		Passo de 70cm	Passo de 60cm	Passo de 50cm
BackPocket	P1	10.980	10.884	10.692
BackPocket	P2	11.172	10.884	10.788
BackPocket	P3	11.076	10.884	10.692
BackPocket	P4	11.076	10.980	10.500
Hand	P1	11.556	11.172	10.500
Head	P1	11.268	10.692	10.692
Pocket	P1	11.268	10.884	10.692
Pocket	P2	11.556	11.172	10.980
Pocket	P3	11.268	10.884	10.884
Pocket	P4	11.747	11.172	10.980
Texting	P1	10.596	10.692	10.500

Tabela 4 – Limiares de cada configuração para janela de tamanho 17.

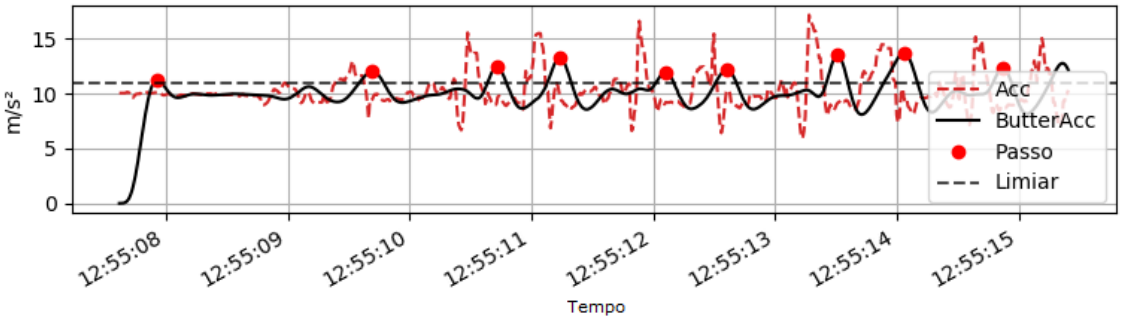
Foi escolhido os valores de limiares do tamanho de passo de $60cm$ por ser um valor intermediário. Para as posições que possuem mais de uma orientação (*Pocket* e *BackPocket*), o limiar foi definido como sendo a média dos valores. Gerando os valores de limiares, descritos na Tabela 5, para cada uma das posições.

Na Figura 49, tem-se a detecção de passo para posição *BackPocket*-P1 com tamanho de passo de $70cm$.

Posição no Corpo	Limiar em m/s^2
BackPocket	10.908
Hand	11.172
Head	10.692
Pocket	11.028
Texting	10.692

Tabela 5 – Limiares escolhidos para cada classe.

Figura 49 – Detecção de passo para a posição *BackPocket*-P1. Trajeto com passo de $70cm$ da base controlada. ButterAcc é a leitura da magnitude do acelerômetro filtrada.



6.4 CLASSIFICADOR DE POSIÇÃO

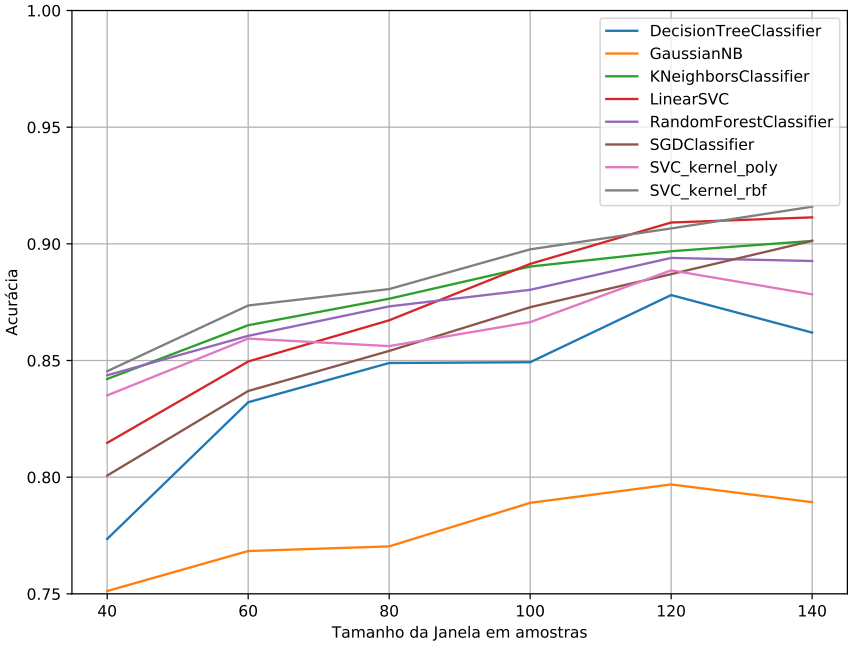
Para classificar a posição do *smartphone*, foram utilizados os classificadores implementados pela biblioteca do Python scikit-learn (PEDREGOSA et al., 2011) versão 0.20.3. Foram testados os seguintes classificadores para esta etapa:

- *K-Nearest Neighbors* (KNN): Na biblioteca scikit-learn, KNeighborsClassifier.
- *Support Vector Machine* (SVM): Na biblioteca scikit-learn, LinearSVC e SVC.
- *Naive Bayes*: Na biblioteca scikit-learn, GaussianNB.
- Árvore de Decisão: Na biblioteca scikit-learn, DecisionTreeClassifier.
- *Random Forest*: Na biblioteca scikit-learn, RandomForestClassifier.
- Regressão Logística: Na biblioteca scikit-learn, SGDClassifier.

Para avaliar os classificadores, foi utilizado o K-Fold estratificado com $K = 5$, utilizando a base controlada (Seção 5.2). Na Figura 50, pode se ver a acurácia dos classificadores com a variação do tamanho da janela (de 40 a 140), considerando-se pra cada janela a melhor configuração de sobreposição dentro de três possíveis (12.5%, 25% e 50%). Os classificadores *K-Nearest Neighbors* (KNN), *Support Vector Machine* (SVM) e Regressão Logística tiveram suas entradas normalizadas de 0 a 1.

Percebe-se uma tendência a uma acurácia maior para tamanhos maiores de janela. Observando-se melhor o comportamento dos quatro melhores classificadores encontrados na etapa anterior, KNN (Figura 51a), SVM (Figura 51b e Figura 51c) e regressão logística (Figura 51d) nota-se que o tamanho da sobreposição da janela não tem um impacto tão grande na acurácia do classificador. Desta forma, será

Figura 50 – Acurácia média dos classificadores utilizando K-Fold com $K = 5$.

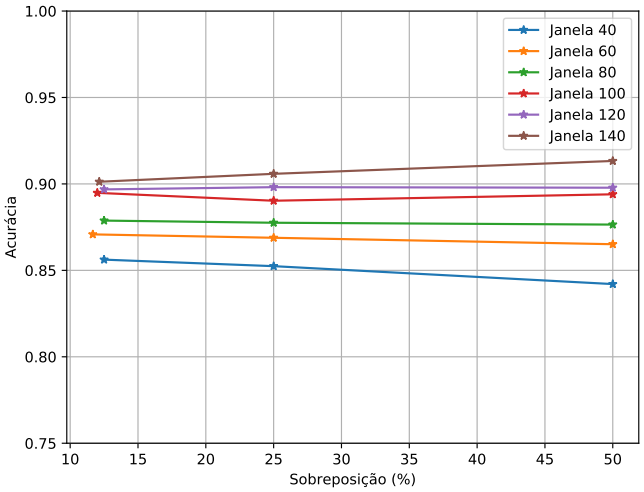


utilizada a janela de tamanho 140 e sobreposição de 50%. Uma vez que esta combinação gera uma carga de processamento menor, a classificação só ocorre uma vez a cada $700ms$.

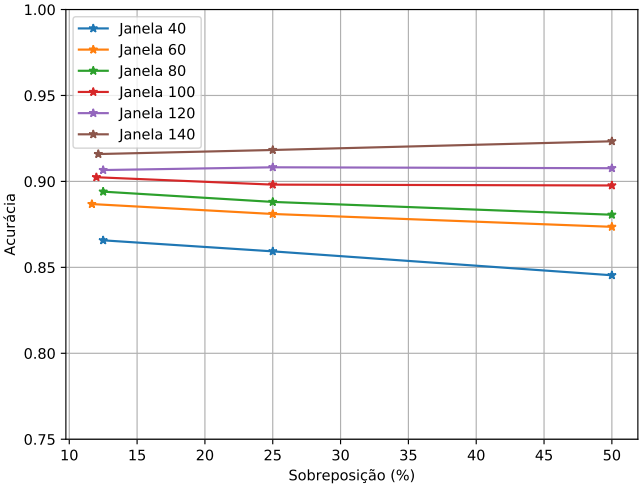
Uma das etapas de processamento do sistema classificador de posição (Seção 4.3) é a redução de dimensionalidade utilizando *Principal Component Analysis*. Desta forma, foi verificada o comportamento da acurácia com a variação do número de componentes principais (Figura 52) para os quatro melhores classificadores.

Percebe-se uma convergência a partir de 25 componentes. Dentre os quatro melhores, foi escolhida a SVM com *kernel* RBF, uma vez que teve um bom resultado e diferentemente do KNN, não existe a necessidade de se armazenar os padrões de treinamento. Na Figura 53, pode-se ver a matriz de confusão para a SVM com *kernel* RBF na base de dados longa não controlada (Seção 5.4).

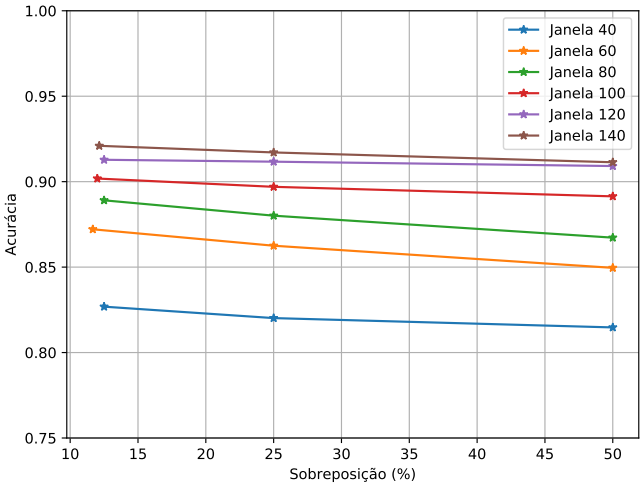
Figura 51 – Acurácia média para diferentes tamanhos de janela variando a sobreposição.



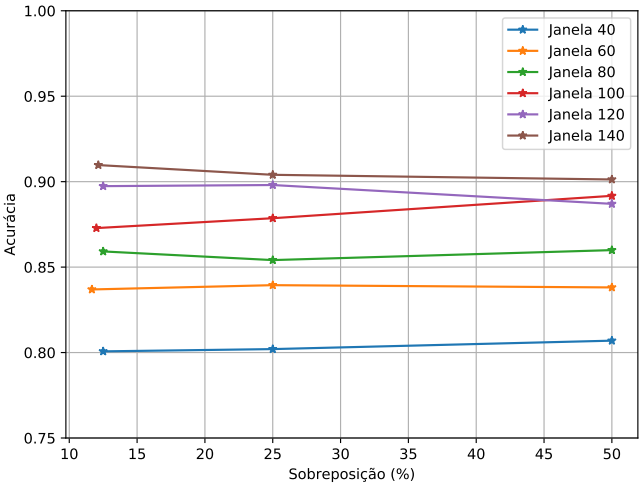
(a) KNN.



(b) SVM com *kernel* RBF.



(c) SVM com *kernel* linear.



(d) Regressão Logística.

Figura 52 – Variação do número de componentes do PCA para os quatro melhores classificadores com tamanho de janela 140 e sobreposição de 50%.

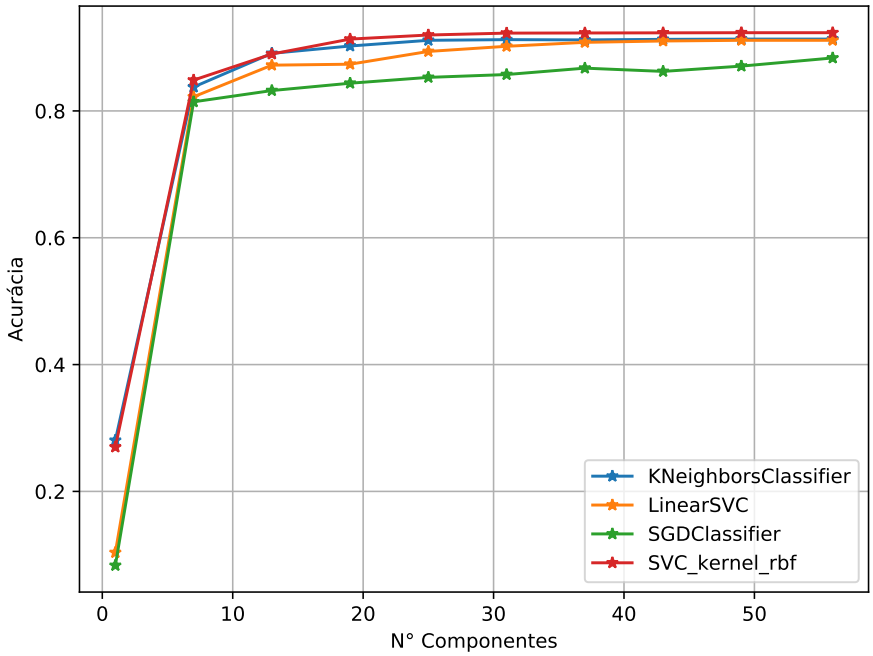
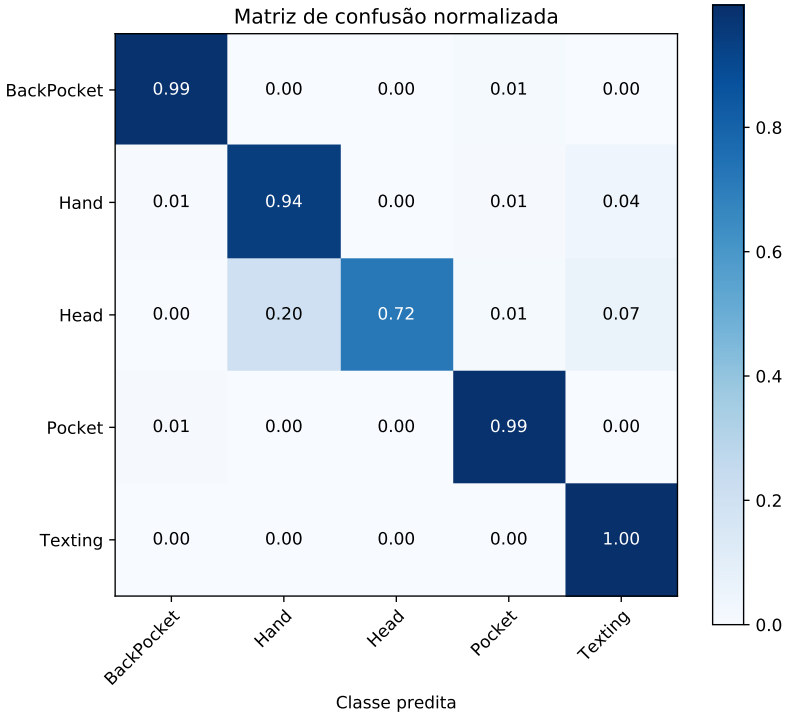


Figura 53 – Matriz de confusão SVM com *kernel* RBF, janela tamanho 140 e sobreposição 50% na base não controlada longa.



6.5 REGRESSOR DE TAMANHO DO PASSO

Para estimar o tamanho do passo do usuário, foram utilizados regressores implementados pela biblioteca do Python scikit-learn (PEDREGOSA et al., 2011) versão 0.20.3, e xgboost (CHEN; GUESTRIN, 2016) versão 0.80. Foram testados os seguintes regressores nesta etapa:

- KNN: Na biblioteca scikit-learn, KNeighborsRegressor.
- SVM: Na biblioteca scikit-learn, LinearSVR e SVR.
- Árvore de Decisão: Na biblioteca scikit-learn, DecisionTreeRegressor.
- *Random Forest*: Na biblioteca scikit-learn, RandomForestRegressor.
- *Extreme Gradient Boosting* (XGBoost): Na biblioteca xgboost, XGBRegressor.
- Regressão Linear: Na biblioteca scikit-learn, LinearRegression.
- Lasso: Na biblioteca scikit-learn, Lasso.
- Ridge: Na biblioteca scikit-learn, Ridge.
- *Elastic Net*: Na biblioteca scikit-learn, ElasticNet.

Para avaliar os regressores, foi utilizado o K-Fold com $K = 5$, utilizando a base controlada (Seção 5.2). O regressor de cada posição foi treinado apenas com dados daquela posição.

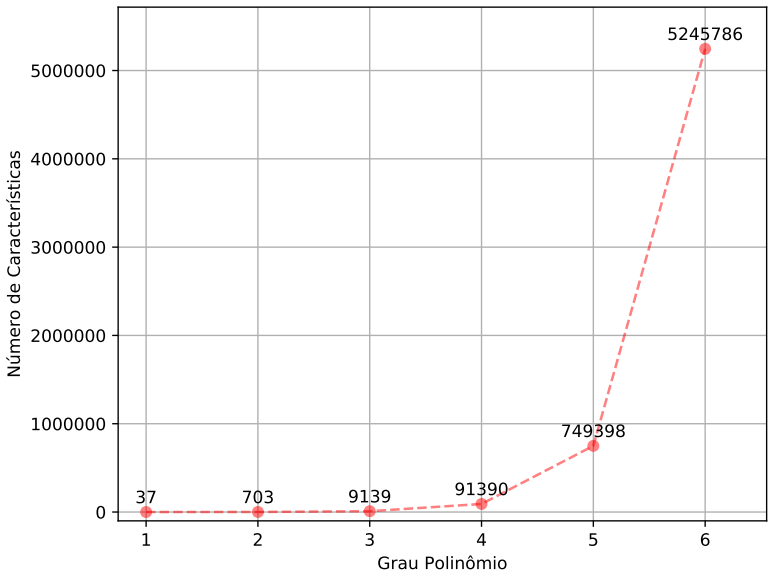
Os regressores KNN, SVM, Regressão Linear, Lasso, Ridge e *Elastic Net* tiveram suas entradas normalizadas de 0 a 1. Para os regressores Regressão Linear, Lasso, Ridge e *Elastic Net* foram geradas características polinomiais na tentativa de se fazer uma regressão polinomial. Os graus de polinômios testados foram 1, 2 e 3. A partir deste valor o número de características geradas é muito grande, como pode ser visto na Figura 54.

6.5.1 *BackPocket*

Na Tabela 6, tem-se a média do RMSE de cada um dos regressores para diferentes tamanhos de janela. Em cada valor de janela tem-se o melhor resultado alcançado por aquele regressor. Analisando a Tabela, tem-se como melhores regressores a Árvore de Decisão, o *Random Forest*, o KNN e o XGBoost.

Dentre os quatro melhores regressores, o KNN foi deixado de fora por ser um regressor baseado em instâncias (faz-se necessário o armazenamento dos padrões de

Figura 54 – Crescimento do número de características para diferentes graus de polinômio.



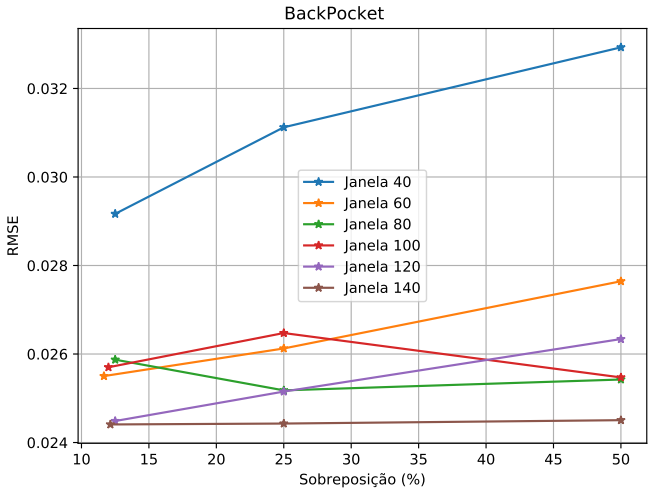
Regressor	Tamanho de Janela					
	40	60	80	100	120	140
DecisionTreeRegressor	0.0292	0.0255	0.0252	0.0255	0.0245	***0.0244***
ElasticNet	0.0427	0.0425	0.0423	0.0421	0.0419	0.0417
KNeighborsRegressor	0.0255	0.0251	0.0247	0.0246	0.0245	***0.0244***
Lasso	0.0427	0.0425	0.0423	0.0421	0.0419	0.0417
LinearRegression	0.0386	0.0350	0.0350	0.0350	0.0354	0.0365
LinearSVR	0.0350	0.0322	0.0305	0.0293	0.0291	0.0292
RandomForestRegressor	0.0275	0.0255	0.0252	0.0251	0.0246	**0.0245**
Ridge	0.0346	0.0321	0.0312	0.0310	0.0310	0.0300
SVR kernel poly	0.1048	0.1047	0.1046	0.1046	0.1045	0.1044
SVR kernel rbf	0.1048	0.1047	0.1046	0.1046	0.1045	0.1044
XGBRegressor	0.0294	0.0263	0.0261	0.0257	0.0252	**0.0250**

Tabela 6 – Valores de RMSE dos regressores treinados com dados da posição *Back-Pocket*. Em negrito, os quatro melhores valores de RMSE dos quatro melhores regressores.

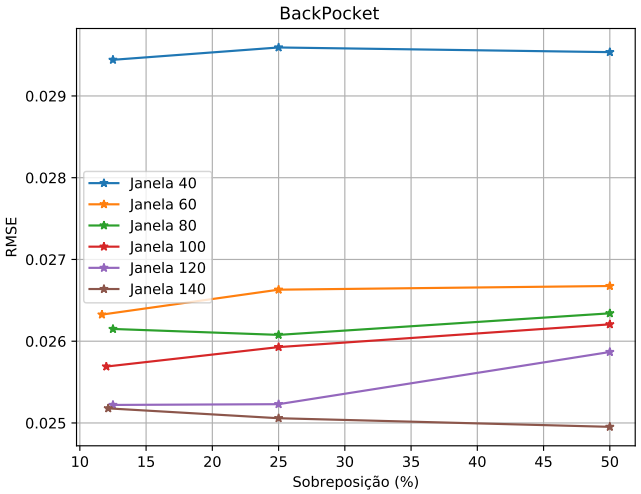
treino). Então sobram a Árvore de Decisão, o *Random Forest* e o *Extreme Gradient Boosting* (XGBoost), todos baseados em árvores.

Nas Figuras 55a, 55b e 55c, tem-se a variação de RMSE para diferentes tamanhos de janela variando a sobreposição. Nota-se, que em maioria, sobreposições menores geram um erro menor.

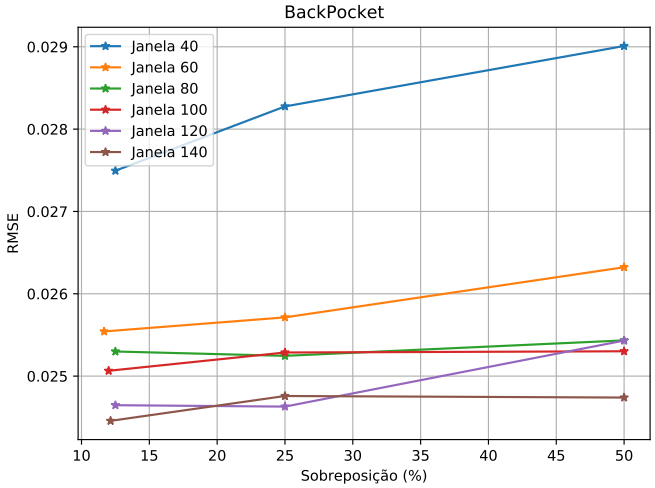
Figura 55 – Valores médios de RMSE para a posição *BackPocket* para diferentes tamanhos de janela, variando a sua sobreposição.



(a) Árvore de Decisão.



(b) XGBoost.



(c) *Random Forest*

.

6.5.2 *Hand*

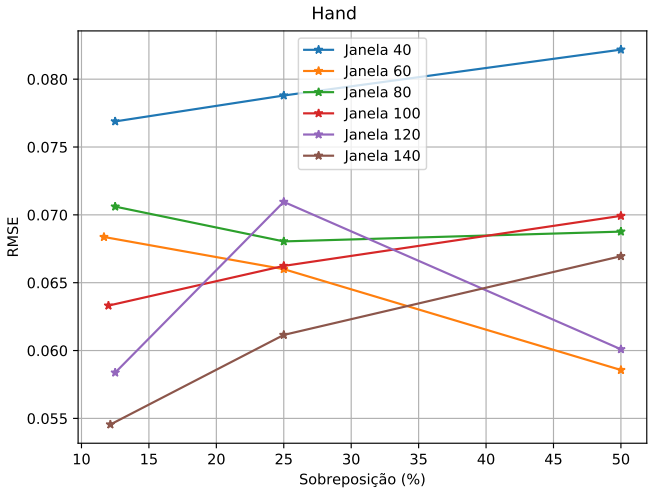
Na Tabela 7, tem-se a média do RMSE de cada um dos regressores para diferentes tamanhos de janela. Em cada valor de janela, tem-se o melhor resultado alcançado por aquele regressor. Analisando a tabela, tem-se como melhores regressores a Árvore de Decisão, o *Random Forest*, o KNN e o XGBoost, assim como no *BackPocket*.

Regressor	Tamanho de Janela					
	40	60	80	100	120	140
DecisionTreeRegressor	0.0769	0.0586	0.0680	0.0633	0.0584	**0.0546**
ElasticNet	0.0806	0.0804	0.0804	0.0803	0.0800	0.0800
KNeighborsRegressor	0.0647	0.0610	0.0634	0.0596	**0.0574**	0.0593
Lasso	0.0806	0.0804	0.0804	0.0803	0.0800	0.0800
LinearRegression	0.2890	0.3068	0.2988	0.2586	0.2014	0.1983
LinearSVR	0.1086	0.1554	0.1534	0.0994	0.1058	0.1287
RandomForestRegressor	0.0612	0.0553	0.0590	0.0566	0.0509	**0.0498**
Ridge	0.1055	0.1114	0.1000	0.0923	0.0851	0.0804
SVR kernel poly	0.1019	0.1019	0.1021	0.1019	0.1021	0.1020
SVR kernel rbf	0.1019	0.1019	0.1021	0.1019	0.1021	0.1020
XGBRegressor	0.0618	0.0570	0.0594	0.0553	0.0522	***0.0463***

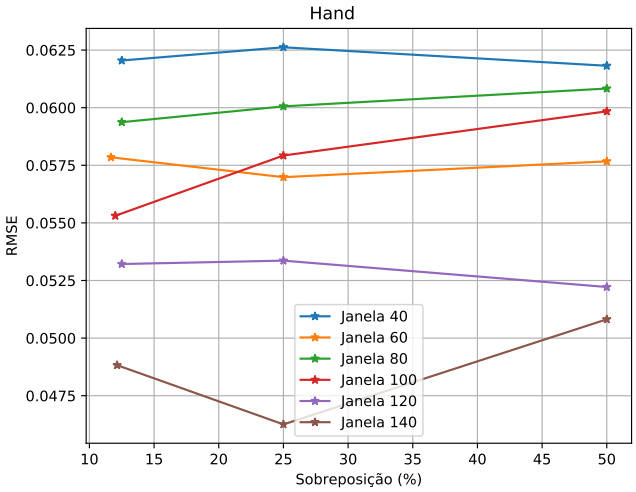
Tabela 7 – Valores de RMSE dos regressores treinados com dados da posição *Hand*. Em negrito, os quatro melhores valores de RMSE dos quatro melhores regressores.

Nas Figuras 56a, 56b e 56c, tem-se a variação de RMSE para diferentes tamanhos de janela variando a sobreposição.

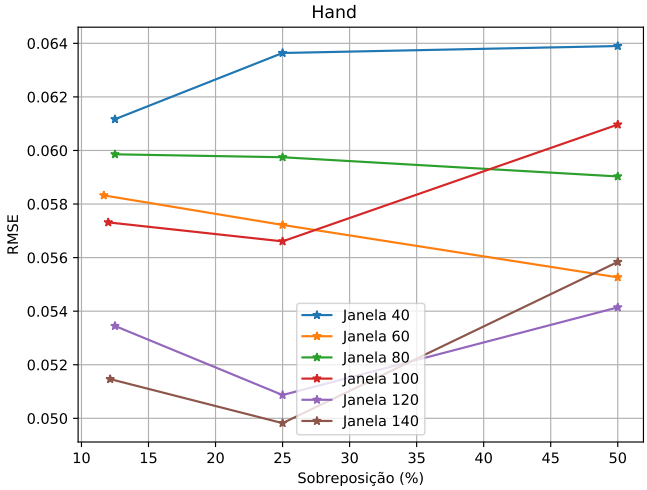
Figura 56 – Valores médios de RMSE para a posição *Hand* para diferentes tamanhos de janela, variando a sua sobreposição.



(a) Árvore de Decisão.



(b) XGBoost.



(c) *Random Forest*.

6.5.3 *Head*

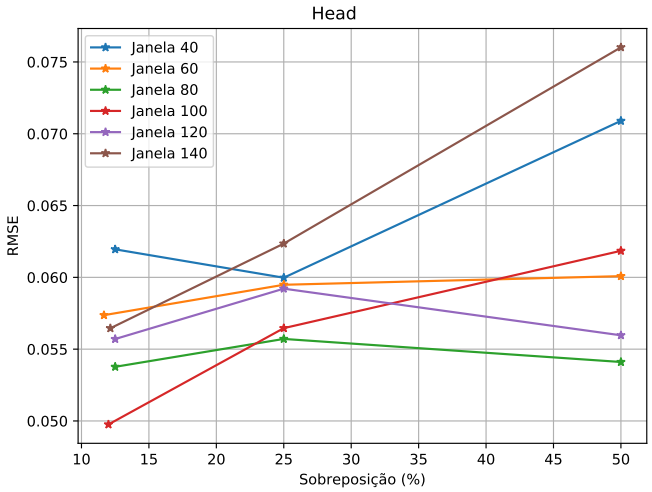
Na Tabela 8, tem-se a média do RMSE de cada um dos regressores para diferentes tamanhos de janela, em cada valor de janela tem-se o melhor resultado alcançado por aquele regressor. Analisando a Tabela, tem-se como melhores regressores a Árvore de Decisão, o *Random Forest*, o KNN e o XGBoost, assim como no *BackPocket* e *Hand*.

Regressor	Tamanho de Janela					
	40	60	80	100	120	140
DecisionTreeRegressor	0.0600	0.0574	0.0538	**0.0498**	0.0557	0.0565
ElasticNet	0.0810	0.0810	0.0808	0.0805	0.0806	0.0803
KNeighborsRegressor	0.0551	0.0524	0.0522	0.0492	0.0505	***0.0490***
Lasso	0.0810	0.0810	0.0808	0.0805	0.0806	0.0803
LinearRegression	0.0798	0.0786	0.0779	0.0721	0.0722	0.0778
LinearSVR	0.0791	0.0726	0.0735	0.0700	0.0676	0.0680
RandomForestRegressor	0.0574	0.0550	**0.0510**	0.0529	0.0567	0.0544
Ridge	0.0707	0.0641	0.0660	0.0647	0.0637	0.0627
SVR kernel poly	0.1019	0.1019	0.1020	0.1019	0.1021	0.1020
SVR kernel rbf	0.1019	0.1019	0.1020	0.1019	0.1021	0.1020
XGBRegressor	0.0572	0.0562	0.0541	**0.0539**	0.0584	0.0549

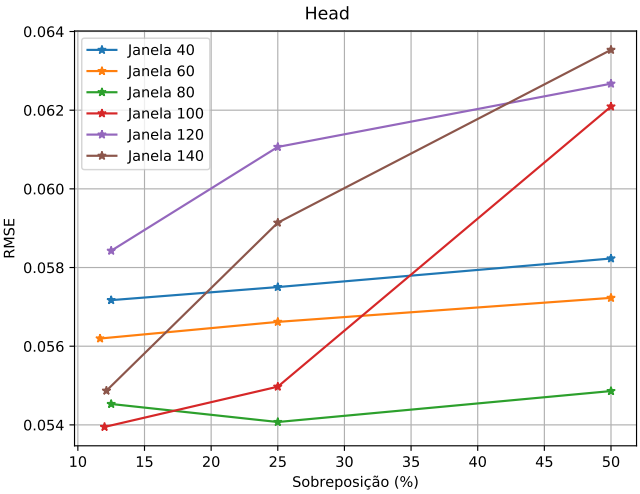
Tabela 8 – Valores de RMSE dos regressores treinados com dados da posição *Head*. Em negrito, os quatro melhores valores de RMSE dos quatro melhores regressores.

Nas Figuras 57a, 57b e 57c, tem-se a variação de RMSE para diferentes tamanhos de janela variando a sobreposição.

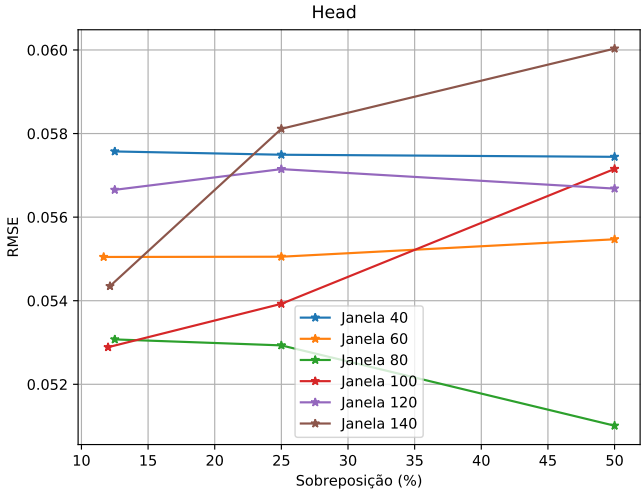
Figura 57 – Valores médios de RMSE para a posição *Head* para diferentes tamanhos de janela, variando a sua sobreposição.



(a) Árvore de Decisão.



(b) XGBoost.



(c) *Random Forest*

.

6.5.4 *Pocket*

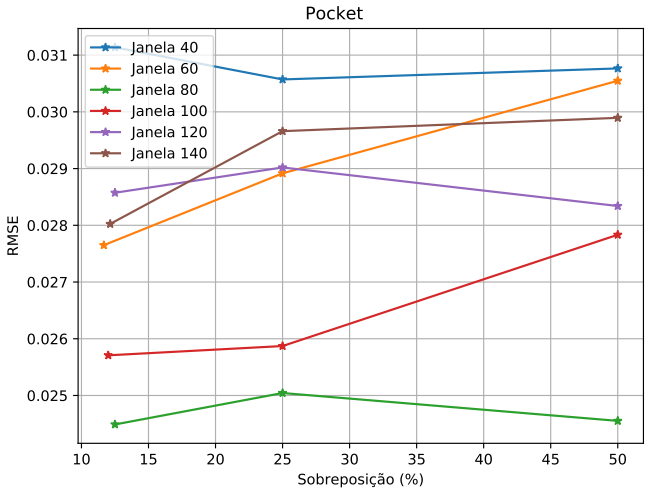
Na Tabela 9, tem-se a média do RMSE de cada um dos regressores para diferentes tamanhos de janela, em cada valor de janela tem-se o melhor resultado alcançado por aquele regressor. Analisando a Tabela, tem-se como melhores regressores a Árvore de Decisão, o *Random Forest*, o KNN e o XGBoost, assim como no *BackPocket*, *Hand* e *Head*.

Regressor	Tamanho de Janela					
	40	60	80	100	120	140
DecisionTreeRegressor	0.0306	0.0276	**0.0245**	0.0257	0.0283	0.0280
ElasticNet	0.0424	0.0422	0.0420	0.0418	0.0416	0.0413
KNeighborsRegressor	0.0265	0.0256	0.0259	0.0255	0.0251	***0.0242***
Lasso	0.0424	0.0422	0.0420	0.0418	0.0416	0.0413
LinearRegression	0.0357	0.0342	0.0333	0.0334	0.0339	0.0342
LinearSVR	0.0314	0.0298	0.0286	0.0276	0.0277	0.0272
RandomForestRegressor	0.0283	0.0268	**0.0250**	0.0259	0.0280	0.0273
Ridge	0.0336	0.0318	0.0313	0.0305	0.0300	0.0298
SVR kernel poly	0.1047	0.1046	0.1045	0.1044	0.1043	0.1042
SVR kernel rbf	0.1047	0.1046	0.1045	0.1044	0.1043	0.1042
XGBRegressor	0.0280	0.0271	0.0263	**0.0261**	0.0267	0.0267

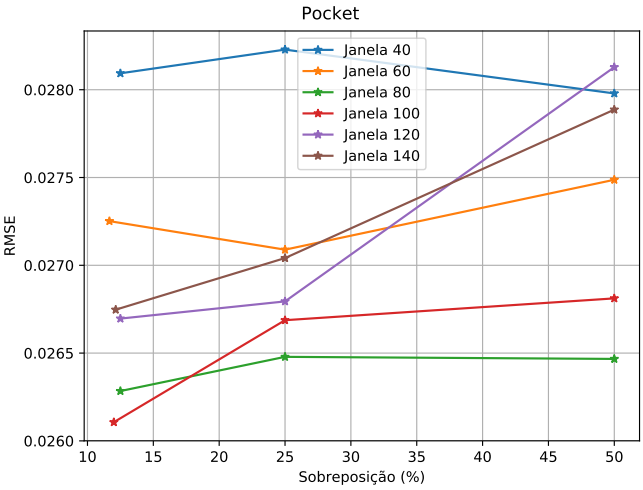
Tabela 9 – Valores de RMSE dos regressores treinados com dados da posição *Pocket*. Em negrito, os quatro melhores valores de RMSE dos quatro melhores regressores.

Nas Figuras 58a, 58b e 58c, tem-se a variação de RMSE para diferentes tamanhos de janela variando a sobreposição.

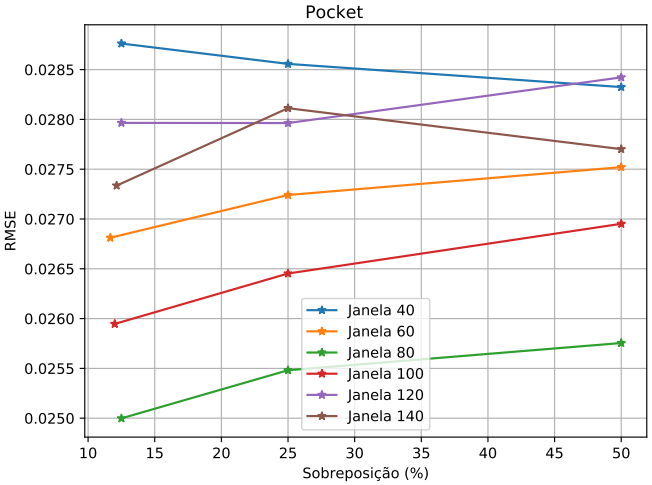
Figura 58 – Valores médios de RMSE para a posição *Pocket* para diferentes tamanhos de janela, variando a sua sobreposição.



(a) Árvore de Decisão.



(b) XGBoost.



(c) *Random Forest*

.

6.5.5 *Texting*

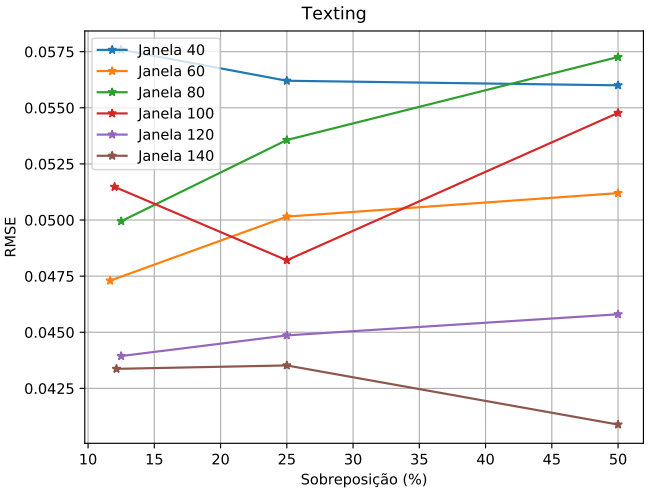
Na Tabela 10, tem-se a média do RMSE de cada um dos regressores para diferentes tamanhos de janela, em cada valor de janela tem-se o melhor resultado alcançado por aquele regressor. Analisando a Tabela, tem-se como melhores regressores a Árvore de Decisão, o *Random Forest*, o KNN e o XGBoost, assim como no *BackPocket*, *Hand*, *Head* e *Pocket*.

Regressor	Tamanho de Janela					
	40	60	80	100	120	140
DecisionTreeRegressor	0.0560	0.0473	0.0499	0.0482	0.0439	**0.0409**
ElasticNet	0.0605	0.0603	0.0600	0.0599	0.0595	0.0594
KNeighborsRegressor	0.0479	0.0423	0.0403	0.0393	0.0384	***0.0377***
Lasso	0.0605	0.0603	0.0600	0.0599	0.0595	0.0594
LinearRegression	0.0614	0.0563	0.0564	0.0557	0.0525	0.0523
LinearSVR	0.0537	0.0517	0.0509	0.0502	0.0510	0.0504
RandomForestRegressor	0.0485	0.0452	0.0471	0.0461	0.0425	**0.0386**
Ridge	0.0554	0.0529	0.0515	0.0508	0.0491	0.0478
SVR kernel poly	0.1115	0.1114	0.1113	0.1113	0.1111	0.1111
SVR kernel rbf	0.1115	0.1114	0.1113	0.1113	0.1111	0.1111
XGBRegressor	0.0482	0.0440	0.0461	0.0453	0.0432	**0.0394**

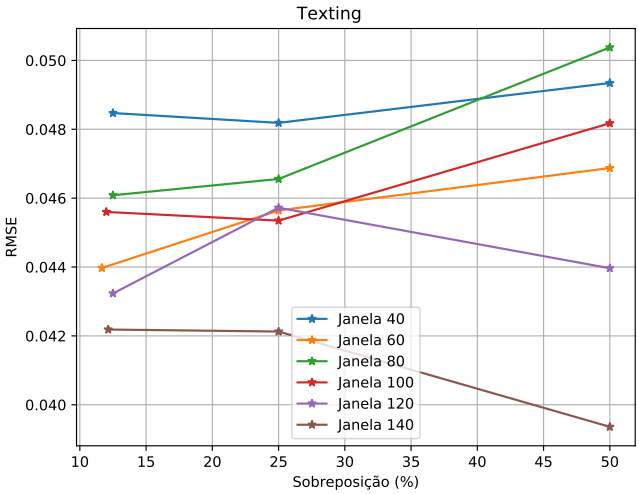
Tabela 10 – Valores de RMSE dos regressores treinados com dados da posição *Texting*. Em negrito, os quatro melhores valores de RMSE dos quatro melhores regressores.

Nas Figuras 59a, 59b e 59c, tem-se a variação de RMSE para diferentes tamanhos de janela variando a sobreposição.

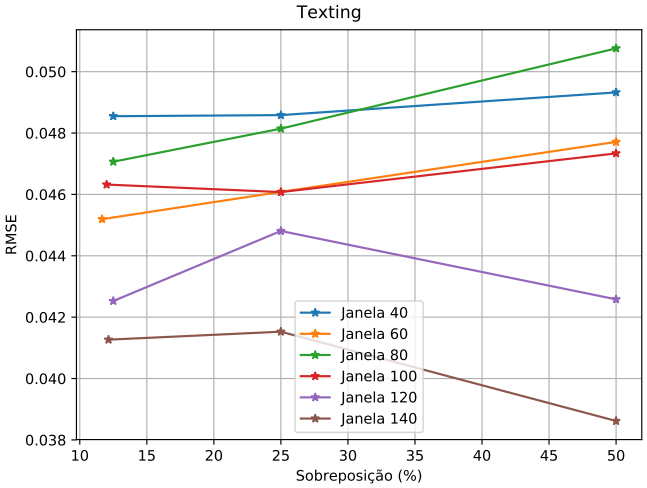
Figura 59 – Valores médios de RMSE para a posição *Texting* para diferentes tamanhos de janela, variando a sua sobreposição.



(a) Árvore de Decisão.



(b) XGBoost.



(c) *Random Forest*

.

6.5.6 Definição Janela

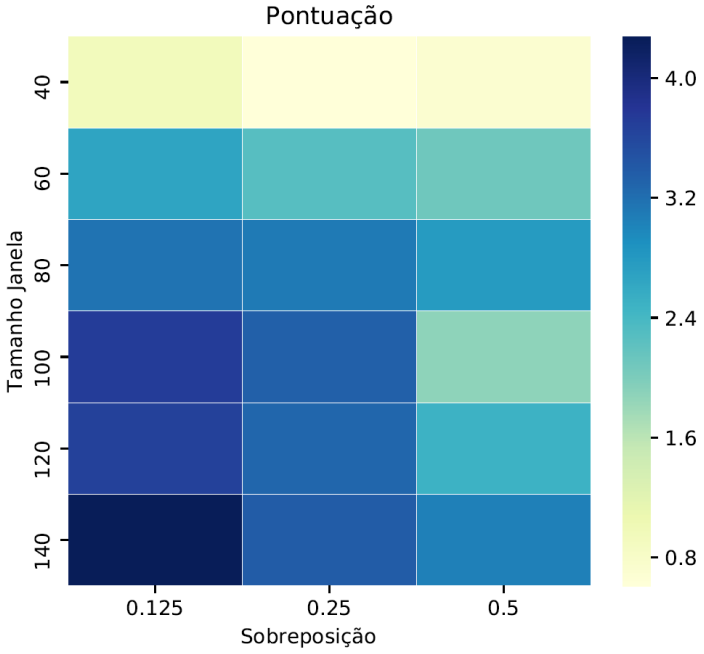
Devido aos variados comportamentos encontrados pelos tamanhos de janela dos melhores regressores de cada posição, foi criado um ranqueamento entre as combinações de janela e sobreposição. Este ranqueamento tem o intuito de definir uma configuração de janela comum a todos, uma vez que a arquitetura proposta na Seção 4.5 possui uma mesma janela para todos os regressores.

O ranqueamento foi construído através do seguinte processo: para cada posição de *smartphone*, foi criada uma pontuação para cada par $(tamanho\,janela,\ sobreposição)$ e essa pontuação é somada ao final, ou seja, cada par vai ter **cinco** pontuações.

A pontuação do par para cada posição é calculada da seguinte forma: cada par recebe como valor de RMSE o menor valor encontrado dentro dos três regressores considerados (Árvore de Decisão, *Random Forest*, e XGBoost, o KNN foi desconsiderado por ser baseado em instância). Os pares então são ordenados pelo RMSE e o par de menor RMSE recebe pontuação máxima e o de maior RMSE pontuação mínima. Os valores de pontução são definidos como sendo o intervalo de 0 a 1 dividido pela quantidade de pares (18 neste caso).

Efetuado este processo tem-se o mapa de calor da Figura 60. A configuração de janela escolhida foi a com tamanho de janela 140 e sobreposição de 12.5%. No quadro 13, tem-se os regressores definidos em cada posição.

Figura 60 – Mapa de calor da pontuação de cada janela.



Posição	Regressor
BackPocket	DecisionTreeRegressor
Hand	XGBRegressor
Head	RandomForestRegressor
Pocket	XGBRegressor
Texting	RandomForestRegressor

Quadro 13 – Regressores escolhidos por posição, utilizando janela com tamanho 140 e sobreposição de 12.5%.

6.6 CLASSIFICADOR DE ORIENTAÇÃO

Para classificar a orientação do *smartphone* foram utilizados os mesmos classificadores da etapa de detecção de posição: KNN, SVM, *Naive Bayes*, Árvore de Decisão, *Random Forest* e Regressão Logística. As variações de janela utilizadas também foram as mesmas, tamanho da janela de 40 a 140, e sobreposição com valores de 12.5%, 25% e 50%.

Para avaliar os classificadores, foi utilizado também o K-Fold estratificado com $k = 5$, utilizando a base controlada (Seção 5.2). O sistema descrito na Seção 4.6 considera apenas duas posições *BackPocket* e *Pocket*, e no treinamento de cada uma apenas os padrões da sua posição são considerados.

6.6.1 *BackPocket*

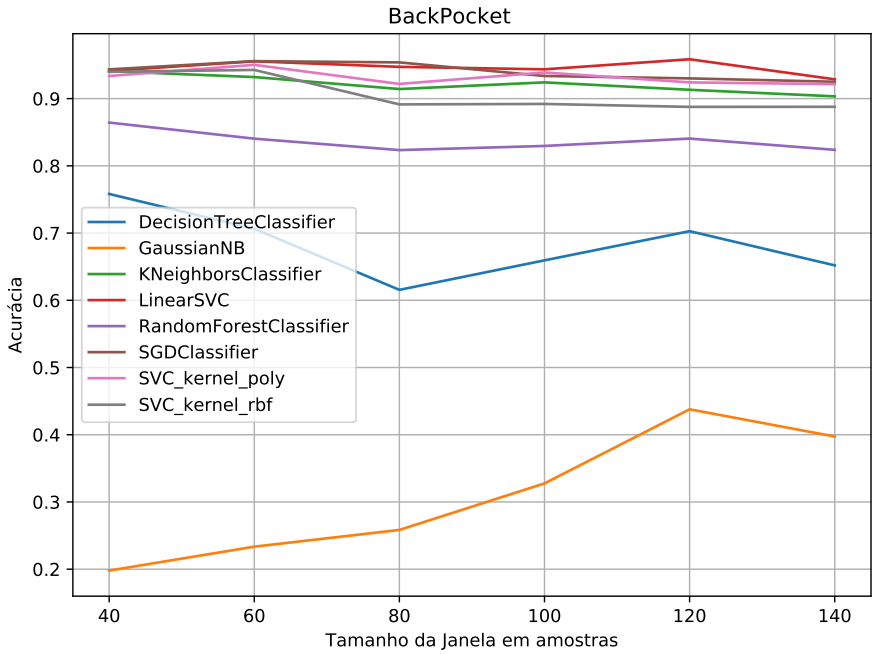
Na Figura 61 pode se ver a acurácia dos classificadores para a posição *BackPocket* com a variação do tamanho da janela, considerando-se para cada janela a melhor configuração de sobreposição dentro das três possíveis.

Os melhores classificadores nesta posição foram KNN, SVM com *kernel* linear, SVM com *kernel* polinomial, e a Regressão Logística. Uma análise mais profunda pode ser vista nas Figuras 62a, 62b, 62c e 62d. A SVM e a Regressão Logística tiveram resultados muito bons, uma vez que algumas combinações passaram dos 0.95 de acurácia.

Dentre esses três melhores percebe-se um resultado muito bom para a janela de tamanho 60. Entretanto, a SVM com *kernel* linear teve um resultado muito bom com uma janela de tamanho 120 também, e como janelas maiores são mais desejadas por necessitar de um processamento menos frequente, a janela de tamanho 120 com sobreposição de 50% foi escolhida para ser melhor analisada.

Na Figura 63, tem-se a acurácia dos quatro melhores classificadores variando-se o número de componentes do PCA. A SVM de kernel linear tem o melhor resultado e a partir de 35 componentes percebe-se sua convergência. O KNN não foi selecio-

Figura 61 – Acurácia média dos classificadores utilizando K-Fold com $k = 5$.



nado para esta análise por se tratar de um classificador baseado em instância (ter que armazenar o conjunto de treino para sua execução).

6.6.2 *Pocket*

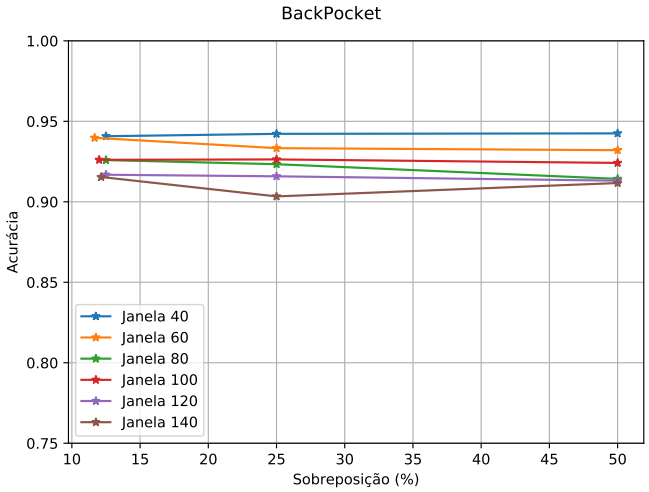
Na Figura 64, pode se ver a acurácia dos classificadores para a posição *Pocket*.

Os melhores classificadores nesta posição foram, SVM com *kernel* linear, SVM com *kernel* polinomial,SVM com *kernel* RBF e a Regressão Logística. Uma análise mais profunda pode ser vista nas Figuras 65a, 65b, 65c e 65d. A SVM com *kernel* linear e a Regressão Logística tiveram resultados muito bons uma vez que algumas combinações ficaram muito perto acurácia máxima. Como o sistema descrito na Seção 4.6 utiliza a mesma janela para ambos os classificadores e a janela de tamanho 120 da SVM de *kernel* linear teve um resultado muito bom, esta janela foi escolhida para ser melhor analisada.

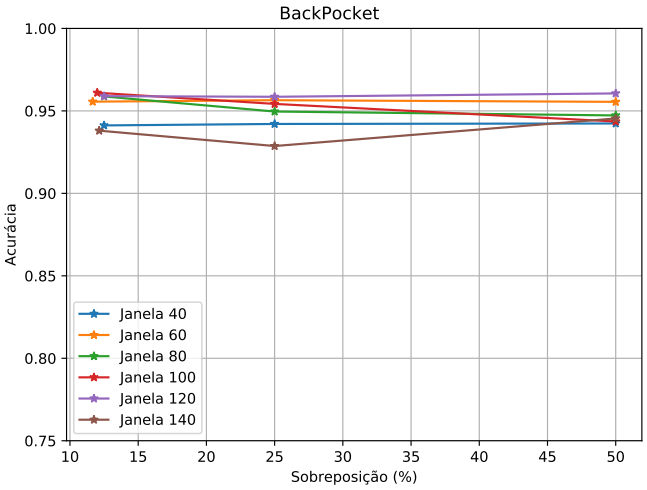
Na Figura 66, tem-se a acurácia dos quatro melhores classificadores variando-se o número de componentes do PCA. A SVM de kernel linear tem o melhor resultado e a partir de 25 componentes percebe-se sua convergência.

Nas Figuras 67a e 67b, tem-se a matriz de confusão para a base não controlada longa, utilizando SVM com *kernel* linear.

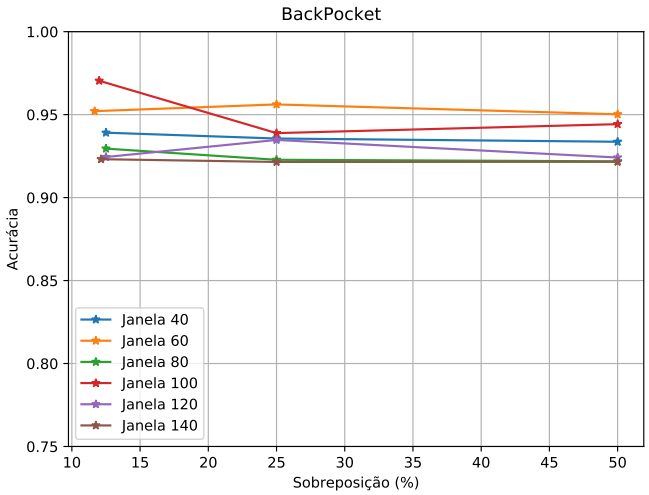
Figura 62 – Acurácia média para diferentes tamanhos de janela variando a sobreposição.



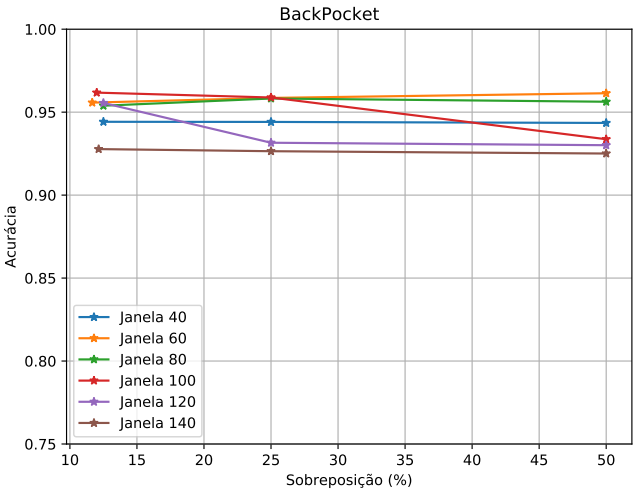
(a) KNN.



(b) SVM com *kernel* linear.



(c) SVM com *kernel* polinomial.



(d) Regressão Logística.

Figura 63 – Variação do número de componentes do PCA para os quatro melhores classificadores com tamanho de janela 120 e sobreposição de 50%.

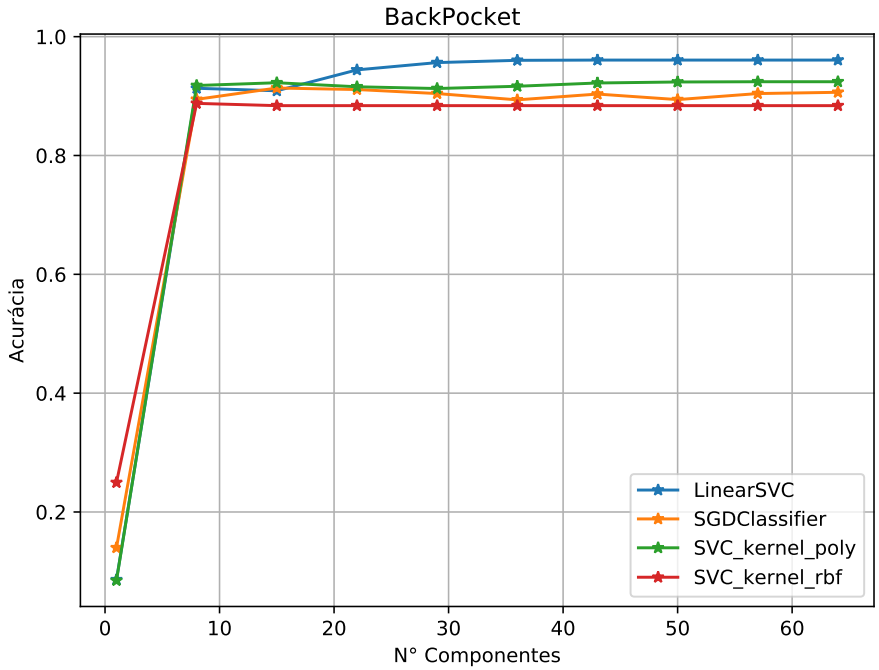


Figura 64 – Acurácia média dos classificadores utilizando K-Fold com $k = 5$.

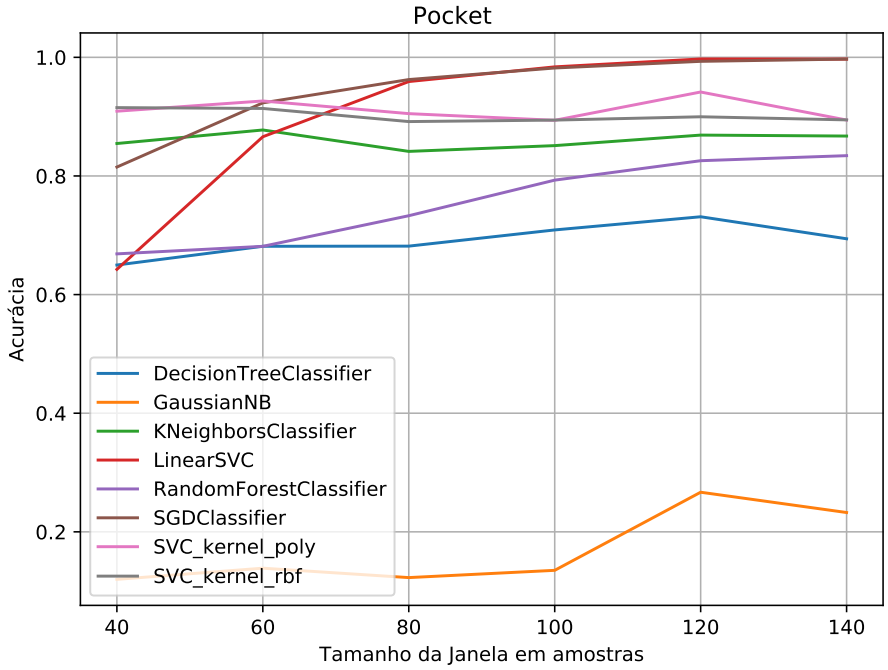
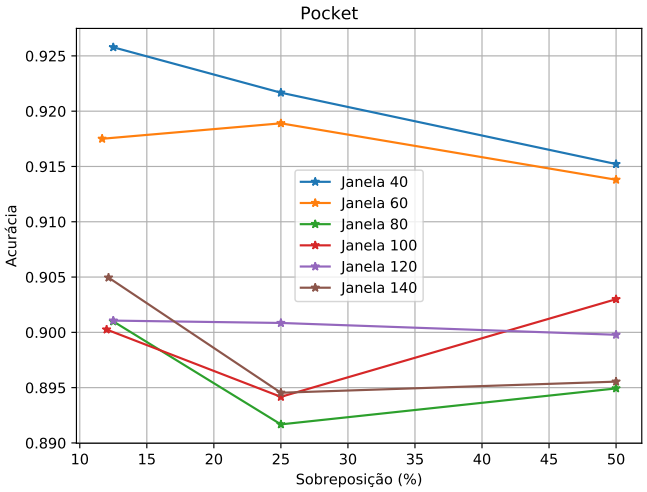
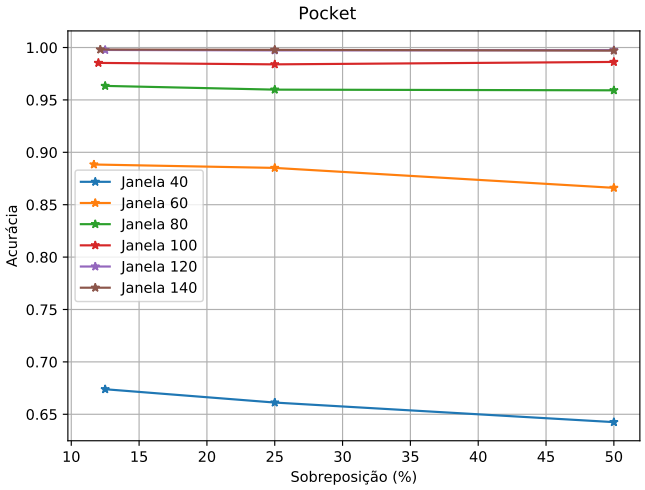


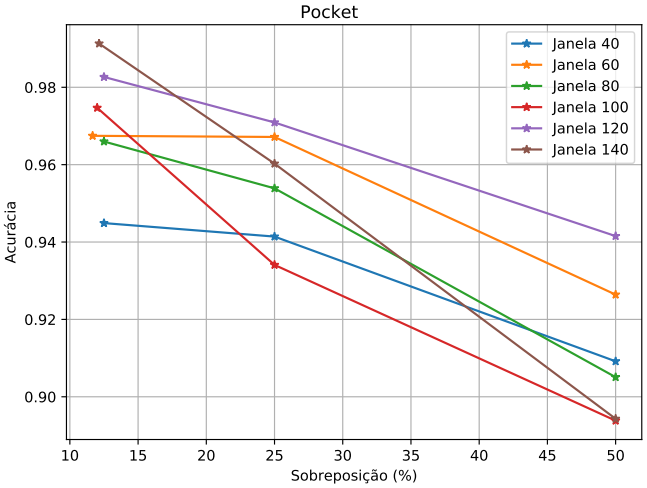
Figura 65 – Acurácia média para diferentes tamanhos de janela variando a sobreposição.



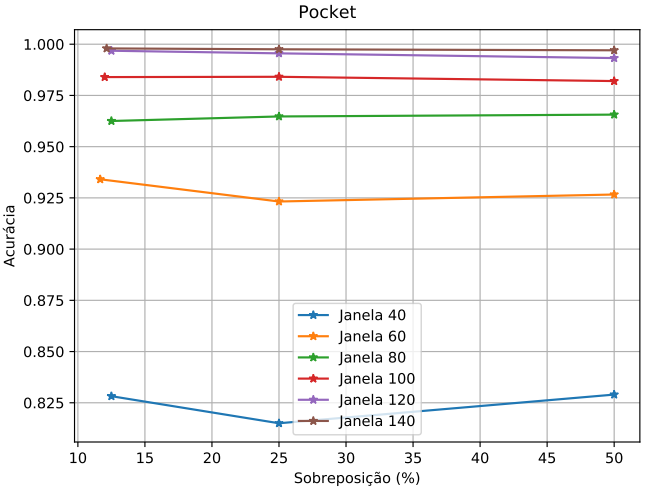
(a) SVM com *kernel* RBF.



(b) SVM com *kernel* linear.



(c) SVM com *kernel* polinomial.



(d) Regressão Logística.

Figura 66 – Variação do número de componentes do PCA para os quatro melhores classificadores com tamanho de janela 120 e sobreposição de 50%.

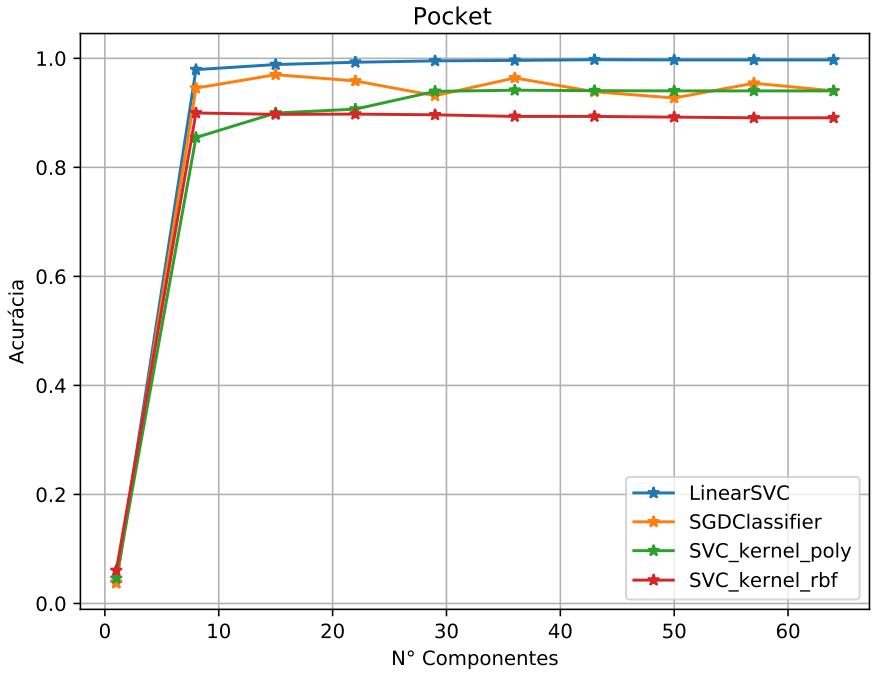
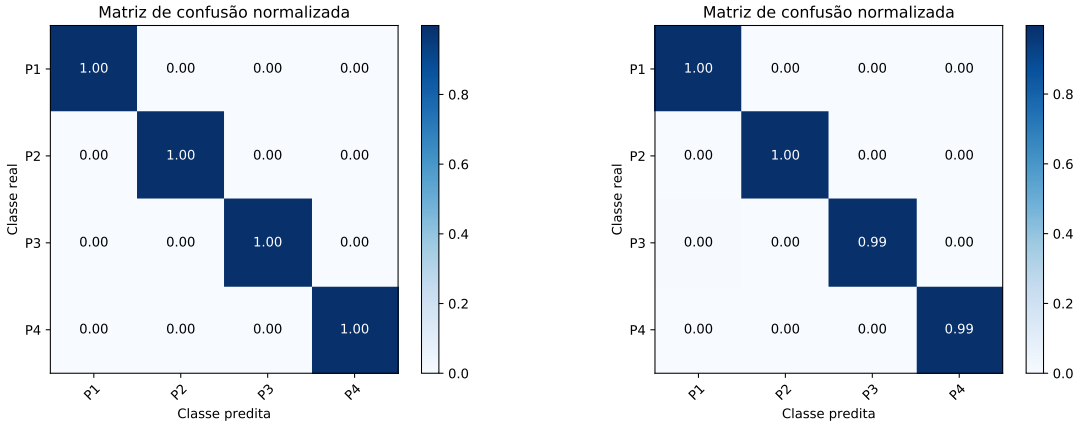


Figura 67 – Matriz de confusão SVM com *kernel* linear, janela tamanho 120 sobreposição 50% na base não controlada longa.



(a) *BackPocket*.

(b) *Pocket*.

6.7 INTEGRAÇÃO DO SISTEMA - LOCALIZAÇÃO 2D

Integrando todos os classificadores e regressores do sistema e utilizando as bases não controladas longa e curta, tem-se respectivamente os resultados das tabelas 11 e 12. Nas Tabelas 11 e 12, tem-se para cada combinação de posição e orientação de *smartphone* o tamanho do trajeto, o tamanho do trajeto predito, o erro absoluto do tamanho trajeto (módulo da diferença do tamanho do trajeto com o tamanho predito), e o erro de localização 2D (distância do ponto de origem para o ponto de término).

Posição	Orientação	Tamanho Trajeto (m)	Tamanho Trajeto Predito (m)	Erro Absoluto Tamanho Trajeto (m)	Erro localização 2D (m)
Texting	-	1019.79	1223.98	204.20	59.40
Hand	-	1084.03	1245.81	161.78	155.00
Head	-	1083.49	1254.63	171.14	170.68
Pocket	P1	3428.14	3886.33	458.20	147.93
Pocket	P2	3630.25	3857.40	227.14	2105.89
Pocket	P3	3574.24	3910.59	336.35	642.50
Pocket	P4	3001.73	3184.99	183.26	1475.72
BackPocket	P1	3498.41	3811.45	313.03	102.14
BackPocket	P2	3532.51	3874.93	342.42	204.39
BackPocket	P3	3445.36	3890.03	444.67	21.41
BackPocket	P4	2831.43	3005.07	173.63	967.75

Tabela 11 – Resultado do sistema de localização 2D utilizando a base não controlada longa.

Os erros de localização gerados pelos trajetos grandes se devem a um *offset* que não foi tratado pelo sistema que detecta a direção (Seção 4.7), fazendo com que este erro acumulado levasse a erros grandes. Estes erros acumulados podem ser vistos nas Figuras 68a e 68b, em que os eixos representam as coordenadas de localização 2D.

Nos trajetos curtos, Figuras 69a e 69b, esse comportamento é muito menor, por isso, o erro de localização é menor.

Quanto à estimação do comprimento do trajeto, os erros normalizados de cada trajeto podem ser vistos na Tabela 13. O erro normalizado é definido como sendo

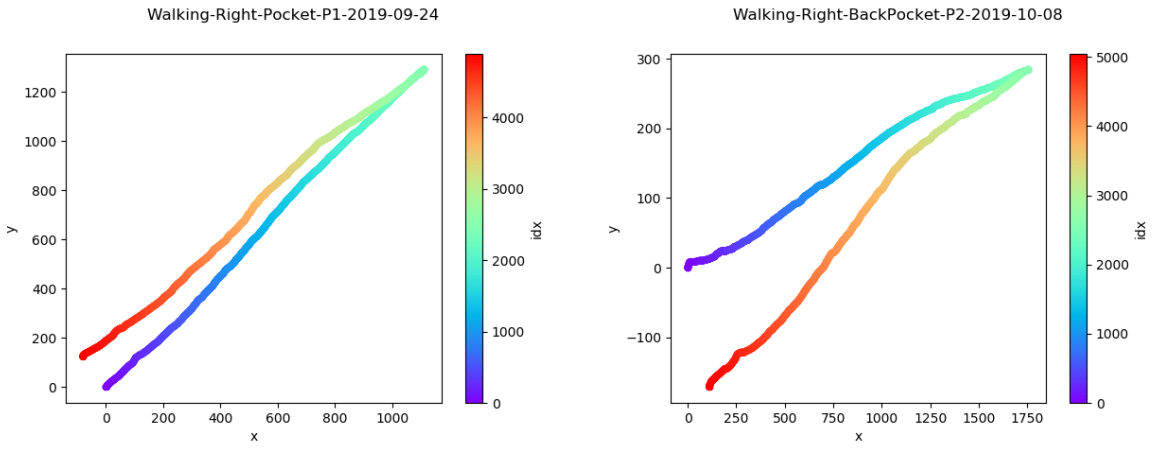
$$\frac{Erro\,Absoluto\,Tamanho\,Trajeto\,(m)}{Tamanho\,Trajeto\,(m)}$$

. Assim, pode-se comparar os erros dos trajetos longos e curtos. Os trajetos longos no geral apresentam erros maiores do que os trajetos curtos, o que se justifica

Posição	Orientação	Tamanho Trajeto (m)	Tamanho Trajeto Predito (m)	Erro Absoluto Tamanho Trajeto (m)	Erro localização 2D (m)
Texting	-	14	13.03	0.97	1.25
Hand	-	14	13.64	0.36	0.64
Head	-	14	13.87	0.13	10.06
Pocket	P1	14	13.96	0.04	1.11
Pocket	P2	14	15.37	1.37	1.91
Pocket	P3	14	14.65	0.65	1.11
Pocket	P4	14	14.62	0.62	1.31
BackPocket	P1	14	15.31	1.31	1.97
BackPocket	P2	14	15.30	1.30	1.99
BackPocket	P3	14	13.87	0.13	0.91
BackPocket	P4	14	17.44	3.44	2.50

Tabela 12 – Resultado do sistema de localização 2D utilizando a base não controlada curta.

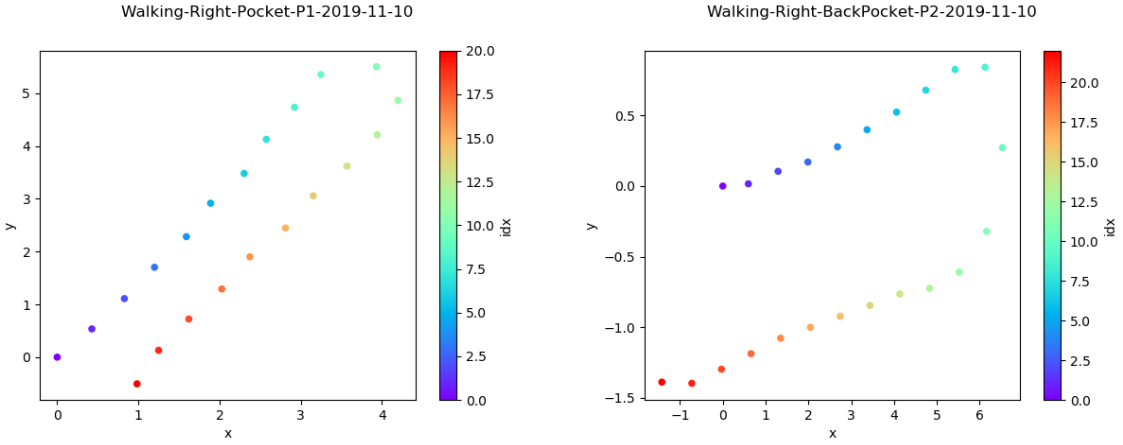
Figura 68 – Exemplo de localização 2D gerada para as posições *Pocket* e *BackPocket*, no trajeto não controlado longo. A barra de *idx* serve para indicar o tempo, *idx* 0 indica a primeira amostra.



(a) *Pocket*. (b) *BackPocket*.

pelo erro acumulado. Uma possível fonte desses erros acumulados é a detecção de passo, porém nos trajetos não controlados, não se sabe o número total de passos.

Figura 69 – Exemplo de localização 2D gerada para as posições *Pocket* e *BackPocket,* no trajeto não controlado curto. A barra de *idx* serve para indicar o tempo, *idx* 0 indica a primeira amostra.



(a) *Pocket.* (b) *BackPocket.*

Posição	Orientação	Erro Normalizado Trajeto Longo	Erro Normalizado Trajeto Curto
Texting	-	0.17	0.07
Hand	-	0.13	0.03
Head	-	0.14	0.01
Pocket	P1	0.12	0.01
Pocket	P2	0.06	0.10
Pocket	P3	0.09	0.05
Pocket	P4	0.06	0.04
BackPocket	P1	0.08	0.09
BackPocket	P2	0.09	0.09
BackPocket	P3	0.11	0.01
BackPocket	P4	0.06	0.25

Tabela 13 – Erros normalizados para predição do tamanho do trajeto.

7 CONCLUSÃO

Este trabalho avaliou diversas técnicas de aprendizagem de máquina que foram empregadas no sistema proposto no Capítulo 4 que possui três principais etapas: detecção de passo, estimação de passo e estimação de orientação.

Para as etapas de estimação de passo e detecção de orientação, faz-se necessário a detecção da posição do *smartphone*. Para isto, foram testados diversos classificadores de naturezas diferentes, baseados em instâncias, em árvores, estatísticos e *Support Vector Machine* (SVM). Nesta etapa, a SVM se mostrou a melhor opção, tendo maior acurácia na base de teste e utilizando janela de 140 amostras com sobreposição de 50%.

Na etapa de estimação de passo, foram utilizados regressores para fazer a estimação do tamanho do passo, um para cada posição do corpo considerada. Foram testados regressores de naturezas diferentes: baseados em instâncias, em árvores, regressão polinomial e SVM.

Notou-se que os regressores mais eficientes foram os baseados em árvores e utilizavam janelas grandes (140 amostras), porém com baixa sobreposição (12,5%). O sistema teve um erro médio normalizado de 10% na base de dados não controlada longa e 7% na base não controlada curta.

Na etapa estimação de orientação, utilizou-se um classificador para identificar a orientação do *smartphone*, uma vez que é necessário para calcular a direção do usuário. Foram consideradas quatro possíveis orientações para o *smartphone*. Estas variações de orientação foram consideradas apenas para as regiões do bolso do usuário, bolso da frente e de trás.

Percebeu-se que o problema era linearmente separável uma vez classificadores como SVM com *kernel* linear teve acurácia perto dos 100%. Foram testados os mesmo tipos de classificadores que na detecção de posição. Para esta etapa também foi utilizada uma janela grande (120 amostras) com alta sobreposição (50%).

O desempenho geral do sistema para localização 2D mostrou-se apenas satisfatório na base não controlada curta. A causa se deve a um erro de *drift* acumulado, não considerado pelo sistema, que degrada bastante a eficiência para trajetos longos.

7.1 CONTRIBUIÇÕES

O presente trabalho apresenta as seguintes contribuições:

- Geração de uma base de dados com cenário controlado.
- Geração de uma base de dados com cenário não controlado com trajetos longos e curtos.
- Desenvolvimento de aplicativo Android para coleta da base de dados.
- Avaliação de técnicas de aprendizagem de máquina para estimação de tamanho de passo e detecção de posição e orientação *smartphones*.
- Sistema de PDR para ser utilizado no contexto de chave veicular.

7.2 DIFICULDADES

As dificuldades encontradas neste trabalho foram ausência de base de dados pública comum para ser utilizada. Cada artigo cria sua própria base de dados com cenários variados, taxa de amostragem diferentes, e com variação nos dados coletados. Falta de detalhamento de parte dos algoritmos utilizados nos artigos, dificultando ou impossibilitando sua replicação.

7.3 TRABALHOS FUTUROS

A partir dos resultados obtidos e do método proposto, alguns direcionamentos podem ser feitos para trabalhos futuros:

- Construção de algum mecanismo de correção de *drift* para melhorar a acurácia da localização 2D em trajetos longos.
- Implementação do sistema em aplicação Android para avaliar outras métricas como consumo de bateria.
- Plugar o sistema proposto em algum mecanismo de chave passiva para controlar quando o serviço de abertura estará funcionando, para de fato inibir os ataques de *relay*.

REFERÊNCIAS

AHAMED, H.; ALAM, I.; ISLAM, M. Handwritten digit recognition system based on lrm and svm algorithm. In: . [S.l.: s.n.], 2019.

ANDROID. *Sensores de movimento*. 2019. <https://developer.android.com/guide/topics/sensors/sensors_motion>. Acessado: 25-11-2019.

BOSER, B. E.; GUYON, I. M.; VAPNIK, V. N. A training algorithm for optimal margin classifiers. In: *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*. New York, NY, USA: ACM, 1992. (COLT '92), p. 144–152. ISBN 0-89791-497-X. Disponível em: <http://doi.acm.org/10.1145/130385.130401>.

CHEN, T.; GUESTRIN, C. Xgboost. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, ACM Press, 2016. Disponível em: <http://dx.doi.org/10.1145/2939672.2939785>.

ELHOUSHI, M.; GEORGY, J.; NOURELDIN, A.; KORENBERG, M. J. Motion mode recognition for indoor pedestrian navigation using portable devices. *IEEE Transactions on Instrumentation and Measurement*, v. 65, n. 1, p. 208–221, 2016. ISSN 00189456.

Foody, G. M.; Mathur, A. A relative evaluation of multiclass image classification by support vector machines. *IEEE Transactions on Geoscience and Remote Sensing*, v. 42, n. 6, p. 1335–1343, June 2004. ISSN 1558-0644.

FRANCILLON, A.; DANEV, B.; CAPKUN, S. Relay Attacks on Passive Keyless Entry and Start Systems in Modern Cars. *Network and Distributed System Security Symposium*, p. 431–439, 2011. Disponível em: <http://eprint.iacr.org/2010/332>.

GARCIA, F. D.; OSWALD, D.; KASPER, T.; PAVLIDÈS, P. Lock it and still lose it —on the (in)security of automotive remote keyless entry systems. In: *25th USENIX Security Symposium (USENIX Security 16)*. Austin, TX: USENIX Association, 2016. Disponível em: <https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/garcia>.

HARLE, R. A Survey of Indoor Inertial Positioning Systems for Pedestrians. *IEEE Communications Surveys & Tutorials*, v. 15, n. 3, p. 1281–1293, 2013. ISSN 1553-877X. Disponível em: <http://ieeexplore.ieee.org/document/6407455/>.

HUA, J.; SHEN, Z.; ZHONG, S. We can track you if you take the metro: Tracking metro riders using accelerometers on smartphones. *IEEE Transactions on Information Forensics and Security*, v. 12, n. 2, p. 286–297, Feb 2017. ISSN 1556-6013.

HUANG, B.; QI, G.; YANG, X.; ZHAO, L.; ZOU, H. Exploiting Cyclic Features of Walking for Pedestrian Dead Reckoning with Unconstrained Smartphones. In: *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. New York, NY, USA: ACM, 2016. (UbiComp '16), p. 374–385. ISBN 978-1-4503-4461-6. Disponível em: <http://doi.acm.org/10.1145/2971648.2971742>.

HUNG, C.-H.; BAI, Y.-W.; CHIANG, C.-H.; WU, H.-Y. An improved design of an automatic detection balance tray with an accelerometer, four servo motors, four force-sensitive resistors and an mcu expansion board. In: *2017 IEEE International Conference on Consumer Electronics (ICCE)*. [S.l.: s.n.], 2017. p. 49–50.

JOACHIMS, T. Text categorization with support vector machines: Learning with many relevant features. In: NÉDELLEC, C.; ROUVEIROL, C. (Ed.). *Machine Learning: ECML-98*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998. p. 137–142. ISBN 978-3-540-69781-7.

KARIM, A. H.; SAADOON, M.; ABDULLAH, M. Arabic handwriting word recognition based on scale invariant feature transform and support vector machine. *Iraqi Journal of Science*, v. 60, n. 2, p. 381–387, Feb. 2019. Disponível em: <http://scbaghdad.edu.iq/eijs/index.php/eijs/article/view/543>.

KIM, J.; JANG, H.; HWANG, D.-H.; PARK, C. A step, stride and heading determination for the pedestrian navigation system. *Journal of Global Positioning Systems*, v. 3, p. 273–279, 12 2004.

KLEIN, I.; SOLAZ, Y.; OHAYON, G. Pedestrian Dead Reckoning With Smartphone Mode Recognition. *IEEE Sensors Journal*, v. 18, n. 18, p. 7577–7584, 2018. ISSN 1530-437X.

KUANG, J.; NIU, X.; CHEN, X. Robust pedestrian dead reckoning based on MEMS-IMU for smartphones. *Sensors (Switzerland)*, 2018. ISSN 14248220.

LEONE, A.; RESCIO, G.; SICILIANO, P. Supervised machine learning scheme for tri-axial accelerometer-based fall detector. In: *2013 IEEE SENSORS*. [S.l.: s.n.], 2013. p. 1–4. ISSN 1930-0395.

MARTINELLI, A.; GAO, H.; GROVES, P. D.; MOROSI, S. Probabilistic context-aware step length estimation for pedestrian dead reckoning. *IEEE Sensors Journal*, v. 18, n. 4, p. 1600–1611, feb 2018. ISSN 1530437X. Disponível em: <http://ieeexplore.ieee.org/document/8116621/>.

MASON, S. Vehicle remote keyless entry systems and engine immobilisers: Do not believe the insurer that this technology is perfect. *Computer Law and Security Review*, Elsevier Ltd, v. 28, n. 2, p. 195–200, 2012. ISSN 02673649. Disponível em: <http://dx.doi.org/10.1016/j.clsr.2012.01.004>.

ME, L. S. N. *Picking Car Locks: How To Find the Right Auto Lock Pick Set*. 2018. <http://lockoutservicenearme.com/picking-car-locks-how-to-find-the-right-auto-lock-pick-set/>. Acessado: 16-11-2019.

MITCHELL, T. M. *Machine Learning*. [S.l.]: MCGRAW-HILL UK, 1997.

MODER, T.; WISIOL, K.; HAFNER, P.; WIESER, M. Smartphone-based indoor positioning utilizing motion recognition. In: *2015 International Conference on Indoor Positioning and Indoor Navigation, IPIN 2015*. IEEE, 2015. p. 1–8. ISBN 9781467384025. Disponível em: <http://ieeexplore.ieee.org/document/7346768/>.

MOTORS, T. *Tesla Model 3 | Owner's Guide*. 2019. <hhttps://www.tesla.com/sites/default/files/model_3_owners_manual_north_america_en.pdf>. Acessado: 17-11-2019.

Osuna, E.; Freund, R.; Girosit, F. Training support vector machines: an application to face detection. In: *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 1997. p. 130–136. ISSN 1063-6919.

OURS, J. C. van; VOLLAARD, B. The Engine Immobiliser: A Non-starter for Car Thieves. *Economic Journal*, v. 126, n. 593, p. 1264–1291, 2016. ISSN 14680297.

PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M.; DUCHESNAY, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011.

PRATAMA, A. R.; WIDYAWAN; HIDAYAT, R. Smartphone-based Pedestrian Dead Reckoning as an indoor positioning system. In: *Proceedings of the 2012 International Conference on System Engineering and Technology, ICSET 2012*. IEEE, 2012. p. 1–6. ISBN 9781467323765. Disponível em: <http://ieeexplore.ieee.org/document/6339316/>.

RANGANATHAN, A.; CAPKUN, S. Are We Really Close? Verifying Proximity in Wireless Systems. *IEEE Security Privacy*, p. 1, 2018. ISSN 1540-7993.

SHIN, B.; KIM, C.; KIM, J.; LEE, S.; KEE, C.; KIM, H. S.; LEE, T. Motion Recognition-Based 3D Pedestrian Navigation System Using Smartphone. *IEEE Sensors Journal*, v. 16, n. 18, p. 6977–6989, 2016. ISSN 1530-437X.

SHIN, B.; LEE, J. H.; LEE, H.; KIM, E.; KIM, J.; LEE, S.; CHO, Y.-s.; PARK, S.; LEE, T. Indoor 3D pedestrian tracking algorithm based on PDR using smarthphone. In: *Control, Automation and Systems (ICCAS), 2012 12th International Conference on*. [S.l.: s.n.], 2012. ISBN 9788993215045. ISSN 15987833.

WANG, B.; LIU, X.; YU, B.; JIA, R.; GAN, X. Pedestrian dead reckoning based on motion mode recognition using a smartphone. *Sensors (Switzerland)*, 2018. ISSN 14248220.

WEINBERG, H. *Using the ADXL202 in Pedometer and Personal Navigation Applications*. Norwood, MA, USA, 2002.