TÉCNICO LISBOA

Treinador autónomo e motivacional de apoio a idosos em jogos de exercício físico

Maria Carolina Varandas Roque

Dissertação para obtenção do Grau de Mestre em

Engenharia Electrotécnica e de Computadores

Orientador(es): Prof. Dr. Plinio Moreno López
Prof. José Alberto Rosado dos Santos Vitor

Júri

Presidente: Prof. João Fernando Cardoso Silva Sequeira
Orientador: Prof. Dr. Plinio Moreno López
Vogal: Prof.ª Cláudia Alexandra Magalhães Soares

Janeiro 2021

Declaração

Declaro que o presente documento é um trabalho original da minha autoria e que cumpre todos os requisitos do Código de Conduta e Boas Práticas da Universidade de Lisboa.

À minha família e amigos,

Resumo

Hoje em dia, o mundo depara-se com uma nova realidade, não observada até aqui, o envelhecimento da população. É comum, nesta faixa etária, o declínio das capacidades físicas e cognitivas das pessoas, apenas fruto da idade ou potenciadas por doenças específicas.

A actividade física surge como uma importante ferramenta na manutenção da qualidade de vida dos idosos, pois o seu nível de independência depende da capacidade de se manterem autónomos nas suas actividades diárias. Contudo, estes são pouco adeptos do exercício físico pois têm receio de eventuais lesões, falta-lhes motivação e companhia, e porque existem poucas actividades direccionadas à sua faixa etária.

Algumas estratégias foram criadas ao longo do tempo, no sentido de minimizar este problema, tais como, os *exergames*, os *virtual coaches* e os *robot-assisted training*. Nestas abordagens, os exercícios são projectados para a população sénior, tendo em consideração as suas limitações. Um dos factores de destaque é o *feedback* que estes sistemas fornecem durante as actividades, estimulando e motivando a participação dos idosos.

Neste trabalho, pretende-se adicionar *feedback* motivacional num conjunto de *exergames*. Os dados são obtidos a partir de várias sessões com os idosos a jogarem os exergames e os fisioterapeutas a fornecerem as indicações necessárias. Através da abordagem da Aprendizagem por Demonstração e dos algoritmos kNN, CRF e HCRF, pretende-se gerar um classificador que prevê que *feedbacks* devem ser dados em determinadas situações do jogo.

Apesar dos resultados não terem sido óptimos, consegue-se ter alguma precisão para determinados *feedbacks*. Sugestões são traçadas para trabalhos futuros.

Palavras-chave: idosos, motivação, *exergames*, exercício físico, *feedback*.

Abstract

Nowadays, the world's population is ageing. The downfall of physical and cognitive capabilities is common due to advanced age or specific diseases.

Physical activity emerges as an important tool in the maintenance of the elderly's quality of life, since their independence relies on the capacity to maintain themselves autonomous in their daily routines. However, the senior population are not very keen on physical exercise, for a variety of reasons, such as, fear of injuries, lack of motivation and company, or the shortage of activities for their age group.

Over time, some strategies have been implemented to minimize the lack of commitment demonstrated by the eldery. Exergames, virtual coaches and robot-assisted training are the ones that stand out the most. In these approaches, the physical exercises are projected to the senior population, taking into consideration their limitations and restraints. One of the most important factors is the feedback provided during the activities, which stimulates and motivates their participation.

The aim of this thesis is to add motivational feedback in a set of specific exergames. The data was obtained from several sessions with the elderly playing the exergames, supported by physiotherapists. Through the Learning from Demonstration approach and the kNN, CRF and HCRF algorithms, it is intended to generate a classifier that predicts which feedbacks should be given in certain game situations.

Even though the results were not optimal, the accuracy of some feedbacks was positive. Finally, suggestions are delineated for future work.

Keywords: elderly, motivation, exergames, physical exercise, feedback.

Conteúdo

Lista de Tabelas

Lista de Figuras

Lista de Acrónimos

ACM *Association for Computing Machinery*.

AMRR *Adaptative Mixed Reality Rehabilitation*.

AVC Acidente Vascular Cerebral.

BFGS *Broyden-Fletcher-Goldfarb-Shanno*.

CRF *Conditional Random Fields*.

GPS *Global Positioning System*.

HCRF *Hidden Conditional Random Fields*.

HMM *Hidden Markov Models*.

IEEE *Institute of Electrical and Electronics Engineers*.

IMPACT *Impact and Motivating Physical Activity using Context*.

kNN *k-Nearest Neighbors*.

MEMM *Maximum Entropy Markov Models*.

PAR *Physically Assistive Robotics*.

PC *Personal computer*.

PEA Perturbações do Espectro do Autismo.

PEPE *Portable Exergames Platform for Elderly*.

RAT *Robot-Assisted Training*.

RGS *Rehabilitation Gaming System*.

SAR *Socially Assistive Robotics*.

SNAP *Sensor Network for Active Play*.

Capítulo 1

Introdução

Com o envelhecimento da população mundial, surge uma preocupação acrescida relativamente aos idosos de todo o mundo. Devem ser desenvolvidos esforços no sentido de minimizar as lesões tão características desta faixa etária.

1.1 Motivação

O mundo depara-se com o envelhecimento geral da população, não só nos países desenvolvidos, como também nos países em desenvolvimento [1]. Pela primeira vez na história da humanidade, em 2018, o número de pessoas com mais de 65 anos ultrapassou o número de crianças com idade inferior a 5 anos [2]. Dados obtidos pelo Departamento de Assuntos Económicos e Sociais das Nações Unidas [2] revelam que, em 2019, cerca de 9% da população mundial ultrapassava os 65 anos de idade e estimam que, até 2050, esse número aumente para os 16%. Esse crescimento ainda é mais notório no número de pessoas com idade superior a 80 anos. Em 1990, apenas 54 milhões de pessoas no mundo tinham 80 anos ou mais, enquanto que, em 2019 este número atingiu os 143 milhões de pessoas. Projecta-se que, em 2050, este número aumente para os 426 milhões de pessoas [2]. Com isto, a necessidade de ajudar e apoiar esta faixa etária tornou-se num foco de preocupação e de interesse para os governos do mundo inteiro [1]. Prevê-se também um impacto substancial nos sistemas de saúde [3] e, por isso, devem ser tomadas medidas no sentido de melhorar a qualidade de vida dos idosos, minimizando esses impactos.

Como consequência da idade, o corpo humano perde capacidades mentais e motoras que levam a limitações na sua mobilidade, a fragilidades acrescidas e a outros problemas de saúde. Marin *et al.* [1] referem que as principais causas que levam ao declínio das capacidades dos idosos estão associadas a complicações como a instabilidade corporal, problemas de equilíbrio e AVCs, que atingem essencialmente a população mais idosa. A reabilitação que normalmente é necessária na existência de um destes problemas, implica a execução repetitiva de exercícios, de modo a que seja possível recuperar as capacidades motoras perdidas. No entanto, esta repetição constante dos exercícios afecta a motivação dos pacientes e, consequentemente, a sua recuperação. A inactividade e a pouca mobili-

dade dos idosos também contribuem para a perda das suas capacidades funcionais [3]. Neste sentido, é essencial a adopção de um estilo de vida saudável, à qual está associada a prática de exercício físico. De acordo com Larsen *et al.* [3], a actividade desportiva consegue efectivamente reduzir a fraqueza e a perda da condição física nos idosos.

O declínio das capacidades mentais e motoras dificulta a prática de exercício físico nesta faixa etária. Todavia, esta não constitui a única barreira existente que afasta os idosos da prática desportiva. A ela junta-se a falta de interesse, a saúde debilitada, o medo de lesões, a falta de companhia, a existência de poucas actividades direccionadas para a sua faixa etária, e a falta de apoio nas deslocações até aos locais apropriados ao exercício físico [3].

Com o objectivo de motivar a população para a prática de exercício físico, foram surgindo várias abordagens ao longo do tempo, entre as quais, os *exergames*. Este tipo de jogos necessita da execução de movimentos físicos para serem realizados. Deste modo, pretende-se superar algumas barreiras na prática desportiva, sobretudo porque a torna mais apelativa, é direccionado às várias faixas etárias e pode ser realizado em ambiente familiar. Larsen *et al.* [3] revelam um impacto positivo destes jogos na condição física dos idosos. Além disso e segundo a mesma fonte, o *feedback* que é dado durante os jogos é um dos factores que os torna mais motivadores. Segundo Marin *et al.* [1] também se têm feito esforços no sentido de aplicar jogos interactivos nos exercícios de reabilitação, tendo sido revelado um impacto positivo nos utilizadores mais idosos, conseguindo aumentar a sua motivação e empenho na reabilitação.

1.1.1 *Feedback* e a sua importância motivacional

O *feedback* dado durante os jogos consiste em mensagens que são transmitidas aos jogadores, com o objectivo de ajudá-los a melhorar o seu desempenho e incentivá-los a não perder o interesse no jogo. Estas mensagens tanto podem dar informação sobre o que deve ser corrigido, dar ênfase ao que se está a fazer bem, como também podem conduzir os jogadores à obtenção de melhores resultados. Assim sendo, é permitido ao jogador auto-avaliar o seu progresso, podendo corrigir a sua prestação e contribuindo, em simultâneo, para a construção da sua confiança na actividade [4, 5].

Lamoth *et al.* [6] efectuaram um estudo comparativo entre o desempenho dos idosos num jogo com e sem *feedback*. Verificaram que nos jogos sem *feedback*, a *performance* dos utilizadores foi sempre pior. Além disso, os idosos consideraram os jogos sem *feedback* mais aborrecidos, dando preferência àqueles que tinham *feedback* em tempo real.

Durante um jogo, o *feedback* pode-se apresentar de diversas formas, seja através da acumulação de pontos à medida que se atingem certas conquistas, breves declarações quando se executam determinadas acções, barras de progresso acompanhando o desempenho do jogador, moedas virtuais que funcionam como recompensas, ou até mesmo um resumo no fim do jogo sumarizando a prestação do jogador [5, 7].

Como é expectável, há determinados *feedbacks* que se revelam mais eficientes e úteis do que outros, consoante o jogo onde estão a ser aplicados. Esta ideia é realçada por Drummond *et al.* [7] e

Burgers *et al.* [8], que também referem que o *feedback* é mais eficiente quando é relacionado com o jogo e não com o jogador, ou seja, elogiar ou culpar o utilizador é menos eficaz do que alertar para o seu desempenho na tarefa. O *feedback* pré-programado foca-se essencialmente no desempenho no jogo e, por isso, obtêm-se melhores resultados comparativamente ao *feedback* verbal que possa ser dado durante a actividade e que poderá focar-se mais no jogador. Além disso, Johnson *et al.* [4] e Burgers *et al.* [8] mostram que, na maioria das vezes, quando o *feedback* proporciona uma explicação de como atingir a resposta correta, obtêm-se melhores resultados do que indicar apenas se o desempenho do jogador está correto ou não.

Adicionalmente, a forma como o *feedback* é apresentado também influencia o resultado final. O *feedback* pode ser visual, em formato de áudio, pode ser exibido através de um agente ou de um vídeo com a demonstração de uma pessoa, entre outras [4]. Verifica-se que o formato do *feedback* não deve sobrecarregar o jogador em termos de carga cognitiva, ou seja, para um jogo puramente visual, o *feedback* deve ser em áudio, para que o utilizador processe a informação através dos dois canais, visual e verbal [4].

O *feedback* positivo (por exemplo: "Muito bem! Assim é que é!") melhora a motivação e a satisfação do jogador, conseguindo ser mais persuasivo que o *feedback* negativo, uma vez que cria uma afirmação de competência nos jogadores [8]. Ao sentirem-se motivados, os utilizadores geram involuntariamente objectivos superiores para si e, consequentemente, conseguem melhorar o seu desempenho. Nos resultados obtidos por Burgers *et al.* [8], verifica-se que os participantes preferem o *feedback* positivo ao negativo, sendo que os que receberam *feedback* positivo sentiram-se mais competentes e autónomos. No mesmo estudo, também se verifica que os participantes que receberam *feedbacks* positivos são mais propícios de voltarem a jogar, imediatamente a seguir ou num futuro próximo.

Portanto, tudo isto realça o potencial que o *feedback* pode ter na motivação dos jogadores, dependendo da forma como é entregue, da interpretação do utilizador e do quanto está relacionado com o seu comportamento [8]. O aumento da motivação nos jogadores pode aumentar a sua adesão aos exercícios físicos [5].

1.2 Formulação do problema

À medida que se envelhece, o corpo humano sofre um declínio nas suas capacidades. Algumas doenças crónicas, como a obesidade, o Parkinson, a hipertensão, a arritmia e a diabetes, podem forçar os idosos a deixar de ser totalmente independentes e a precisar de auxílio nas suas rotinas diárias, desde tomarem banho a simplesmente mudarem a sua posição corporal, reduzindo a sua qualidade de vida [9].

A actividade física pode ajudar a manter e a melhorar a *performance* física, nomeadamente, em termos de capacidades aeróbicas, de equilíbrio, força e flexibilidade [10]. Todavia, a adesão ao exercício físico convencional é baixa, principalmente se forem exercícios de prevenção. Segundo Baranowski *et al.* [5], quando não existe nenhum benefício imediato, é preciso haver bastante motivação para convencer os idosos a aderir às actividades físicas. Além disso e de acordo com a mesma fonte, os

idosos estão mais abertos às novas tecnologias se perceberem que conseguem retirar algum benefício das mesmas.

Sendo a actividade física tão importante nos idosos e, ao mesmo tempo, com tão pouca adesão por parte dos mesmos, é necessário criar estratégias que os incentivem a realizar tais actividades físicas, de forma voluntária e a longo prazo.

Algumas abordagens foram desenvolvidas ao longo dos últimos anos no sentido de resolver esse problema. Contudo, a maior parte delas baseia-se em ferramentas digitais, tal como iremos analisar no Capítulo 2, e foram projectadas para a população mais jovem. A falta de familiaridade dos idosos relativamente às novas tecnologias representa só por si um obstáculo, uma vez que pode reduzir a sua motivação ou até desenvolver uma eventual frustração durante o processo de aprendizagem [9]. É, por isso, importante que durante o design dos jogos se tenham em consideração as limitações do público-alvo, garantindo assim a acessibilidade dos mais velhos [1].

Uma das características que se destaca no incentivo à actividade física é o carácter motivacional proporcionado pelas abordagens. O *feedback* é uma das característica dos jogos que se salienta na motivação dos seus jogadores [5]. No estudo realizado por Bleakley *et al.* [10], confirma-se que as várias intervenções que foram dadas ao longo dos jogos foram consideradas divertidas, variadas, motivadoras, desafiantes e transmitiam um bom resumo do desempenho do jogador. Todavia, o *feedback* também deve ter em consideração para quem se dirige, contemplando as suas características e limitações.

1.3 Solução proposta

Num conjunto de *exergames* desenhados especificamente para idosos, com o intuito de os tornar fisicamente activos, pretende-se adicionar a *feature* de *feedback* verbal. Estes jogos já possuem *feedback* visual no decorrer dos seus exercícios. Porém, como se tratam de jogos puramente visuais, torna-se difícil a percepção das mensagens por parte dos idosos, que acabam por não conseguir detectar as explicações que lhes estão a ser dadas no ecrã. A adição do *feedback* verbal visa resolver esta falha.

Nesta dissertação são estudados vários modelos de classificação que podem ser utilizados na previsão dos *feedbacks* motivacionais a transmitir aos idosos, em determinados momentos dos jogos.

1.4 Estrutura da Dissertação

A presente dissertação está organizada em 5 capítulos, de acordo com o trabalho desenvolvido.

No Capítulo 1 é descrita a motivação que conduziu à realização e ao desenvolvimento deste trabalho. Explica-se a necessidade de encontrar soluções para ajudar a melhorar a saúde geral de uma população em envelhecimento. Explica-se também a importância do papel motivacional e estimulante desempenhado pelo *feedback*.

O Capítulo 2 contextualiza algumas das soluções encontradas até agora, que visam minimizar o impacto associado ao processo de envelhecimento da população e a falta de motivação para a actividade

física, na terceira idade. Foi realizada uma revisão bibliográfica que engloba diversas tecnologias utilizadas e estudadas até à presente data, para superar este problema da falta de adesão da população mais idosa ao exercício físico. No fim deste capítulo, descreve-se com maior pormenor dois exemplos de sistemas de treino físico, idênticos ao utilizado nesta dissertação.

No Capítulo 3 é referido o ponto de partida para a elaboração deste trabalho e todas as decisões tomadas ao longo do mesmo. Explica-se também a abordagem que se utilizou na sua implementação e os algoritmos usados. Por último, neste capítulo, é descrito de forma sucinta o código MATLAB desenvolvido.

É no Capítulo 4 que são descritos, com detalhe, todos os testes realizados e é feita a análise e discussão dos resultados obtidos. Paralelamente, é efectuada uma comparação entre as várias estratégias utilizadas. No fim do capítulo são apresentadas outras abordagens que poderiam ter sido adoptadas.

Por último, no Capítulo 5 são traçadas algumas conclusões que se puderam retirar, após o desenvolvimento deste trabalho, e são feitas algumas propostas de melhoria, que poderiam ser utilizadas e implementadas num trabalho futuro.

Capítulo 2

Estado da Arte

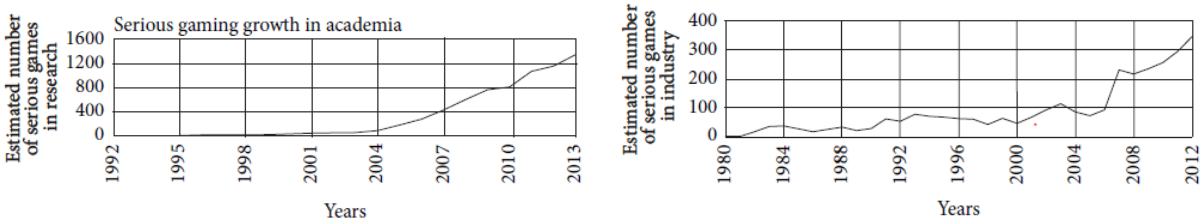
Neste capítulo, descrevem-se as diversas abordagens que foram sendo criadas ao longo dos últimos anos, com o objectivo de motivar as pessoas, particularmente os mais idosos, a aderirem à prática do exercício físico. Ir-se-à abordar os *serious games*, os *Robot-Assisted Training* (RAT) e os *virtual coaches*. No final deste capítulo, são apresentados e explicados 2 exemplos de sistemas robóticos, de forma mais detalhada.

2.1 *Serious games*

O conceito de *serious game* foi primeiramente estabelecido por Clark Abt, em 1970, tendo sido descrito como um jogo cujo objectivo principal é a educação e não o lazer, mas não impossibilitando que esses jogos sejam, ao mesmo tempo, divertidos [11]. Hoje em dia, ainda não existe um conceito claro para os *serious games* [4, 12, 13]. Contudo, a maioria da literatura encontrada sobre o tema concorda que estes jogos têm o propósito da aquisição e consolidação de conhecimentos ou de competências, e não propriamente de entretenimento, o que está de acordo com a ideia inicialmente proposta em 1970. De acordo com Laamarti *et al.* [12], para um *serious game* ser bem sucedido deve equilibrar a parte divertida, característica de um jogo, com o seu objectivo real/"sério", ou seja, o entretenimento nunca deve ser sacrificado. A parte divertida de um qualquer jogo é o que o torna motivador e estimulante para os utilizadores. Portanto, um *serious game* por mais "sério" que seja, deve ser agradável para que o objectivo real possa ser alcançado.

A área dos *serious games* tem crescido significativamente ao longo das últimas décadas, não só no domínio da investigação, como também na indústria de jogos. Laamarti *et al.* [12] efectuaram uma pesquisa de artigos e estudos relacionados com os *serious games*, tendo-se baseado nos arquivos *online* de duas grandes editoras *The Association for Computing Machinery* (ACM) *Digital Library* e *Institute of Electrical and Electronics Engineers* (IEEE) *Xplore Digital Library*. A pesquisa abrange o intervalo de tempo de 1995 até 2013 e os seus resultados apresentam-se na Figura 2.1(a). É possível verificar um crescimento exponencial no número de artigos publicados neste âmbito, o que demonstra o aumento do interesse da sociedade nesta área. Além disso, Laamarti *et al.* [12] também obtiveram

os resultados da Figura 2.1(b), onde se revela um crescimento exponencial semelhante no número de *serious games* na indústria. Contudo, neste caso, o intervalo temporal analisado é superior, abrangendo os anos desde 1980 até 2012.



(a) Crescimento dos *serious games* no domínio da investigação (b) Crescimento dos *serious games* na indústria dos jogos

Figura 2.1: Crescimento dos *serious games* ao longo das últimas décadas. Fonte: Laamarti *et al.* [12].

Com a emergência desta nova indústria, surge a necessidade de estabelecer uma base formal para a mesma e, por isso, em 2002, foi criada a iniciativa *Serious Games* [1, 11], com o intuito de promover o uso dos jogos para fins "sérios". Em 2004, foi fundado o projecto *Games for Health*, cujo principal objectivo é apoiar a comunidade que se dedica aos cuidados de saúde, com novos conhecimentos e desenvolvimentos para a aplicação de jogos nesta área [1, 7]. Fruto deste projecto, é realizada anualmente a conferência *Games for Health* nos Estados Unidos da América e na Europa, onde são apresentados e debatidos temas como o *exergaming*, a fisioterapia, a reabilitação e o exercício físico [1, 14].

Têm sido desenvolvidos *serious games* em diversos domínios, incluindo as áreas da educação, treino, bem-estar, publicidade, património cultural, comunicação interpessoal, cuidados de saúde, entre outras [12]. Na Tabela 2.1 apresentam-se exemplos de *serious games*, que se enquadram em algumas das áreas indicadas anteriormente. Além disso, também são indicadas algumas das características desses jogos. De seguida, será explicado com maior pormenor em que consiste cada um destes jogos.

Tabela 2.1: Classificação de *serious games*. Adaptado de Laamarti *et al.* [12].

Área de aplicação	**Nome do jogo**	**Actividade**	**Modalidade**	**Interacção**	**Individual/ Multijogadores**
Educação	*Skills Arena*	Mental	Áudio/visual	Nintendo Gameboy	Individual
	Making History	Mental	Áudio/visual	Interfaces tradicionais	Individual e Multijogadores
	Lost in the Middle Kingdom	Mental	Áudio/visual	Interfaces tradicionais	Individual
	Roma Nova	Mental	Áudio, visual e háptico	Interfaces tradicionais e dispositivo háptico	Individual
Bem-estar	*Fish 'n' Steps*	Física	Visual	Pedómetro	Individual e Multijogadores
	Sensor Network for Active Play (SNAP)	Física	Áudio/visual	Sensores	Individual e Multijogadores
Assistência médica	*21 Tally*	Mental e física	Áudio/visual	Sensores de biofeedback	Individual
	Rehabilitation Gaming System	Fisiológica	Áudio/visual	Luvas de dados e câmara	Individual

Educação

Como se sabe, hoje em dia, muitos jovens dedicam uma grande parte do seu tempo a jogar videojogos. Assim, se esses jogos tivessem um carácter educacional, estariam a contribuir para a aprendizagem dos seus jogadores. Laamarti *et al.* [12] referem que a natureza viciante dos jogos digitais pode facilitar o processo de aprendizagem dos seus utilizadores.

Alguns jogos educacionais são desenvolvidos para serem utilizados em sala de aula, como por exemplo, o jogo *Skills Arena* e o jogo *Making History* (Tabela 2.1). O primeiro ajuda os estudantes a desenvolver as suas competências aritméticas, possuindo diferentes níveis de dificuldade, e é usado na Nintendo GameBoy [12]. Na Figura 2.2(a) está representada a interface que é apresentada ao jogador. Através do estudo desenvolvido por Shin *et al.* [15] em alunos do 1º ciclo, provou-se que este jogo melhorava efectivamente o desempenho dos estudantes na área da matemática. No que diz respeito ao segundo jogo, este foi usado para avaliar a aprendizagem de conteúdos relacionados com a 2ª Guerra Mundial, em alunos do ensino básico, fazendo a comparação com os métodos de ensino tradicionais. Resultados obtidos por Watson *et al.* [16] revelam que os estudantes se mostravam mais interactivos, mais interessados e com maior motivação durante o processo de aprendizagem.

Por outro lado, também existem jogos educacionais para uso independente, por exemplo, o jogo *Lost in the Middle Kingdom* [17], que ajuda os seus utilizadores a aprenderem uma nova língua, e o jogo *Roma Nova* [18] que, através de um dispositivo háptico, *Novint Falcon*, permite aos jogadores sentirem propriedades 3D de artefactos, tais como, as suas texturas e formas. O projecto *Roma Nova* baseia-se no modelo *Rome Reborn*, que proporciona um ambiente digital tridimensional, que pode ser explorado em tempo real, tal como é possível observar na Figura 2.2(b). O *Rome Reborn* é considerado o modelo mais fidedigno da Roma Antiga existente na actualidade [18].

(a) Ecrã do jogo *Skills Arena*.

(b) Vista do utilizador no jogo *Roma Nova*.

Figura 2.2: Exemplos de *serious games* direccionados para a educação. Fontes: (a) Shin *et al.* [15] e (b) Arnab *et al.* [18].

Bem-estar

Uma vida sedentária contribui para o aparecimento e agravamento de algumas doenças, como a diabetes, a obesidade ou os problemas cardiorrespiratórios, como já foi referido anteriormente. A prática

de exercício físico pode funcionar como meio de prevenção contra estes problemas. Os *serious games* também podem actuar nesse sentido, uma vez que quando direccionados para o bem-estar, têm o objectivo de incentivar os seus utilizadores a serem fisicamente activos, num ambiente de entretenimento e motivador [12].

Um exemplo de um jogo desta natureza é o *Fish 'n' Steps* [19], onde os jogadores utilizam um pedómetro para contar o número de passos dados diariamente. O intuito é ser jogado por um grupo, ou seja, por vários participantes, proporcionando um ambiente competitivo e cooperativo, que contribui para aumentar a motivação dos seus utilizadores. Num estudo efectuado por Lin *et al.* [19], foi possível confirmar que os jogadores desenvolveram padrões de actividades diárias mais saudáveis.

Outro exemplo é o caso do sistema *Sensor Network for Active Play* (SNAP) [20], onde os jogadores têm vários sensores colados em diferentes partes do corpo, de modo a ser possível rastrear todos os seus movimentos. Este sistema possui vários jogos, sendo um deles a recriação de vários movimentos de dança. No estudo efectuado por Whitehead *et al.* [20] mostrou-se que este sistema superava alguns jogos comerciais, como os da Nintendo Wii, em termos de frequência cardíaca e gastos de energia. Além disso, também foi classificado pelos seus participantes como tendo uma alta vertente de diversão.

Assistência médica

Os jogos que fornecem assistência médica dirigem-se não só a pessoas com algum problema de saúde, seja físico ou cognitivo, como também, e de forma geral, a toda a população. Laamarti *et al.* [12] realizaram um enquadramento destes jogos, dividindo-os em 4 grupos distintos: monitorização de saúde, prevenção e educação terapêutica, detecção e tratamento, e reabilitação. Neste sentido, estes jogos podem englobar aqueles que se direccionam para o "bem-estar".

Por exemplo, o jogo *21 Tally* [21], apresentado na Tabela 2.1, insere-se no âmbito da detecção de doenças. Este consiste num conjunto de jogos 2D com o objectivo de detectar a atenção dividida[1], ou seja, impõe que os seus utilizadores prestem atenção a várias dimensões simultaneamente, a fim de averiguar a sua capacidade cognitiva. Na Figura 2.3(a) está apresentado um exemplo de um jogo pertencente à colecção do *21 Tally*.

Por outro lado, a reabilitação é uma das áreas da saúde com maior interesse por parte dos *serious games*, sendo que estes conseguem de facto ser uma boa solução para a falta de motivação dos pacientes. A repetição constante dos mesmos exercícios na reabilitação tradicional leva os seus utilizadores a perderem o interesse e a motivação para os realizar [12]. Por exemplo, o jogo *Rehabilitation Gaming System* (RGS) [22] é um sistema baseado em realidade virtual, que foi projectado para ajudar na recuperação de pacientes com lesões cerebrais. Com o auxílio de umas luvas de dados, que detectam os movimentos dos dedos, e de uma câmara, que acompanha os pulsos e os cotovelos, este sistema pretende estimular as áreas motoras danificadas [22]. Na Figura 2.3(b), é possível observar uma representação do funcionamento deste sistema.

[1] A atenção dividida é uma habilidade cognitiva que permite processar diferentes fontes de informação e responder a vários estímulos em simultâneo. É vital em várias tarefas do dia-a-dia e tende a deteriorar-se com a idade e com algumas doenças [21].

(a) Exemplo de um jogo do *21Tally*.

(b) Representação do RGS.

Figura 2.3: Exemplos de *serious games* direccionados à assistência médica. Fontes: (a) McKanna *et al.* [21] e (b) Cameirão *et al.* [22].

2.1.1 *Exergames*

Os *exergames* podem ser considerados *serious games*, uma vez que estes jogos têm um propósito maior que não apenas o entretenimento. Os *exergames* têm como principal característica a necessidade da execução de diversos movimentos físicos, para serem jogados. Desta forma, são uma alternativa ao exercício físico tradicional. Como a prática de exercício físico é tão importante na sociedade, ajudando a prevenir e a reduzir o risco de doenças crónicas e ainda por outras razões já referidas na secção 1.1, os *exergames* podem ser uma alternativa fiável ao exercício físico em si, contribuindo para melhorar a qualidade de vida e a satisfação da população [9]. Além disso, também podem ajudar a contrariar a pouca adesão dos idosos à actividade física, devido às suas características motivadoras e estimulantes.

Breakley *et al.* [10] realizaram uma revisão à literatura sobre a aplicação dos *exergames* nos idosos, de modo a examinar os efeitos físicos e cognitivos que estes exercícios tinham nos seus utilizadores. Esta pesquisa abrange estudos publicados no intervalo de Janeiro de 2000 até Junho de 2011 e excluíram-se todos aqueles que se direccionavam para a reabilitação, debruçando-se apenas nos que se focavam na melhoria geral do bem-estar. Breakley *et al.* [10] verificaram que estes jogos eram seguros para os idosos e os seus exercícios eram eficazes na melhoria da sua condição física e cognitiva. Contudo, também afirmaram que estas conclusões eram precoces e que seria necessário mais evidências que as suportem, uma vez que apenas 12 estudos preencheram os seus critérios de inclusão.

Posteriormente, Larsen *et al.* [3] também levaram a cabo uma revisão à literatura existente sobre os *exergames*, publicada entre 2011 e 2012, de forma a apurar os seus efeitos na saúde física dos idosos. À semelhança do estudo anterior, também foram excluídos os jogos orientados para a reabilitação e foram apenas considerados aqueles que se dirigiam a idosos saudáveis. Dos sete estudos resultantes, confirmou-se, mais uma vez, que os *exergames* têm potencial no que diz respeito à melhoria da saúde e bem-estar físico dos seus utilizadores, sendo uma forma agradável e motivadora de fazer exercício. Contudo, foi também referido que são necessárias mais pesquisas no sentido de quantificar os efeitos destes jogos nos idosos. Verificou-se que os *exergames* não envolvem apenas actividade física, mas

também exigem trabalho cognitivo, por exemplo, na detecção de estímulos, na atenção e na tomada de decisões rápidas.

Por outro lado, Zeng *et al.* [9] referem que têm sido publicados muitos estudos sobre os *exergames*, mas são poucos os que estão direccionados para a reabilitação da população mais idosa. Este artigo refere que só começaram a existir mais pesquisas sobre os *exergames* a partir de 2008, sendo que a sua aplicação na reabilitação é relativamente recente. Neste estudo, é feita uma revisão dos efeitos dos videojogos na reabilitação de pessoas com idades superiores a 60 anos, com alguma doença crónica ou dificuldade motora, tendo sido consideradas pesquisas publicadas entre Janeiro de 2000 e Agosto de 2016. Zeng *et al.* [9] verificaram a existência de benefícios nos resultados físicos dos seus utilizadores e, mais importante, não se observaram efeitos negativos no uso dos mesmos. Os *exergames* revelaram ser uma aposta vantajosa em relação à reabilitação tradicional pois, para além dos benefícios físicos, também são uma boa forma de entretenimento. O factor mais apelativo destes jogos recai nas suas características motivadoras, levando os seus utilizadores a serem mais participativos e a demonstrarem entusiasmo durante os exercícios de reabilitação [9]. No entanto, a maior parte dos idosos não está familiarizada com as novas tecnologias, pelo que ter de aprender a usar este tipo de jogos pode ser desmotivador e frustrante, principalmente para aqueles que já se sentem afectados pelo tratamento em si. Em suma, Zeng *et al.* [9] consideram os *exergames* como potenciais ferramentas a utilizar na reabilitação dos mais idosos, realçando os seus efeitos positivos tanto na componente física, como na psicológica e cognitiva dos seus utilizadores.

No estudo efectuado por Marin *et al.* [1], foi inicialmente feita uma pesquisa sobre os pontos-chave no *design* de jogos para os idosos. Estes devem ter em consideração as mudanças físicas associadas ao envelhecimento e as doenças mais comuns nestas idades. A segunda fase deste estudo focou-se na procura de tecnologias com uma componente de entretenimento, aplicadas na reabilitação de pacientes que sofreram de AVCs, e de jogos para treinar o equilíbrio dos idosos, uma vez que o AVC e as quedas são os problemas mais comuns nesta faixa etária. Relativamente à primeira fase, foi feita uma revisão da literatura, onde se salientou o trabalho de Flores *et al.* [23]. Este estudo apresenta um conjunto de critérios que é necessário ter em consideração no *design* dos jogos, de modo a proporcionar um ambiente de entretenimento nos idosos. Estes critérios encontram-se sintetizados na Tabela 2.2. Flores *et al.* [23] referem que os jogos devem ser competitivos e desafiantes o suficiente para conseguirem manter o interesse dos seus jogadores, mas tendo, ao mesmo tempo, um objectivo simples. A parte social proporcionada pelos jogos também se destaca, quando estes permitem ser jogados por vários utilizadores. A interface exibida ao utilizador também deve ser tida em atenção, por exemplo, deve-se usar um tamanho da fonte grande e algum contraste para facilitar a visualização dos jogos.

Na segunda fase do estudo de Marin *et al.* [1], foram analisados vários jogos que cumprem estes critérios. Na Tabela 2.3 apresentam-se alguns deles, que tanto podem ser direccionados para a reabilitação, como para treinar o equilíbrio ou mesmo para o exercício físico no geral.

O primeiro jogo da Tabela 2.3 corresponde a uma versão modificada do jogo *Dance Dance Revolution*, desenvolvida por Smith *et al.* [24]. O seu principal objectivo é treinar a locomoção dos idosos, sendo que este constitui um dos problemas mais comuns nesta faixa etária. Neste exercício é apenas

Tabela 2.2: Síntese das características dos jogos para idosos. Adaptado de Flores *et al.* [23].

Critérios para o entretenimento dos idosos
Desafio cognitivo apropriado
Objectivo simples
Interface simples
Feedback motivacional
Componente social
Temas do interesse dos idosos
Adaptado a respostas mais lentas

Tabela 2.3: *Exergames* de reabilitação e de exercício físico. Adaptado de Marin *et al.* [1].

Nome do jogo	**Tecnologia**	**Interacção**	**Característica especial**	***Feedback***	**Histórico de progresso**
Adaptação do jogo *Dance Dance Revolution*	PC e *dancing pad*	Pisar uma plataforma	Música, temática do jogo, partilha do desempenho do jogador com o terapeuta	Visual e Áudio	Sim
Rabbit Chase	PC, câmara e marcador (ex: luvas)	Movimentar as mãos	Mecanismo que adapta a dificuldade do jogo à performance do jogador	Visual e Áudio	Sim
Bubble Trouble					
Arrow Attack					

necessário um monitor (por exemplo, uma televisão ou o monitor de um computador) e um *dancing pad* que corresponde a um tapete de dança com oito setas. No ecrã vão aparecendo as setas que o utilizador deve pisar. Este jogo está adaptado para captar as respostas mais lentas, habituais nos idosos, e foi projectado para ser usado em casa. A característica adicional que os autores propõem é um *design* que monitorize o desempenho dos utilizadores, enviando-o automaticamente aos seus terapeutas. Assim, não é necessário haver um encontro físico entre ambas as partes, uma vez que o profissional consegue acompanhar todos as *performances* dos seus pacientes.

Burke *et al.* [25] desenvolveram um conjunto de jogos, *Rabbit Chase*, *Arrow Attack* e *Bubble Trouble*, também apresentados na Tabela 2.3. O objectivo destes jogos assenta em proporcionar uma ferramenta de reabilitação dos membros superiores, sendo simultaneamente de baixo custo e que pudesse ser usada em casa. Para estes jogos é necessário uma câmara, um computador e um marcador. Este marcador tanto pode ser umas luvas coloridas ou um objecto de uma única cor forte. A cor do marcador deve ser escolhida de modo a não gerar conflitos com a cor do plano de fundo e, nos casos dos jogos que necessitam dos dois braços, cada uma das mãos deve ter uma cor diferente. Estes jogos exigem movimentos dos braços, sendo que a posição das mãos é detectada através do marcador que está a ser usado. O jogo *Rabbit Chase* foi desenvolvido para a reabilitação de um braço apenas (ou o braço direito ou o esquerdo). O segundo jogo, *Arrow Attack*, foi concebido para a reabilitação dos dois braços simultaneamente. No caso do jogo *Bubble Trouble* existem duas variantes, uma versão focada na reabilitação do braço afectado e a outra versão exige o movimento de ambos os braços. O conceito de cada um destes jogos é explicado com mais detalhe nos pontos que se seguem, de acordo com as informações de Marin *et al.* [1] e de Burke *et al.* [25].

Rabbit Chase - Este jogo apresenta quatro buracos e um coelho (ver Figura 2.4). O coelho sai de um dos buracos e entra noutro, sendo que ambos são escolhidos aleatoriamente pelo jogo. O jogador deverá indicar, com a mão, o buraco para onde o coelho se está a dirigir. Caso o utilizador acerte, é fornecido um *feedback* visual e de áudio no sentido de congratular e encorajar o jogador. Além disso, a dificuldade do jogo é ajustada automaticamente conforme o sucesso do utilizador no jogo.

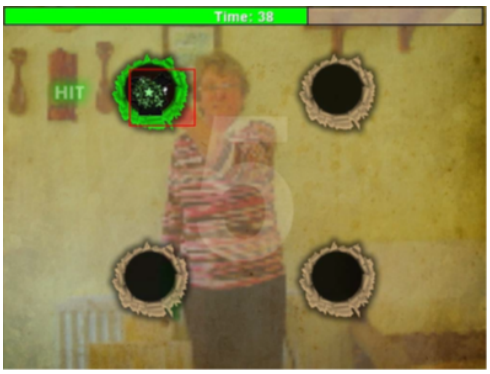


Figura 2.4: Exemplo do jogo *Rabbit Chase*. Fonte: Burke *et al.* [25].

Arrow Attack - Neste jogo são apresentadas quatro caixas onde se movem duas setas, uma aponta para a esquerda e outra para a direita, o que corresponde ao braço esquerdo e ao braço direito, respectivamente (ver Figura 2.5). Além disso, cada seta apresenta-se da cor do marcador que está a ser usado em cada mão, para ajudar o utilizador a distinguir qual dos braços usar. O objectivo do jogo é tocar simultaneamente em cada uma das setas com a mão correta.

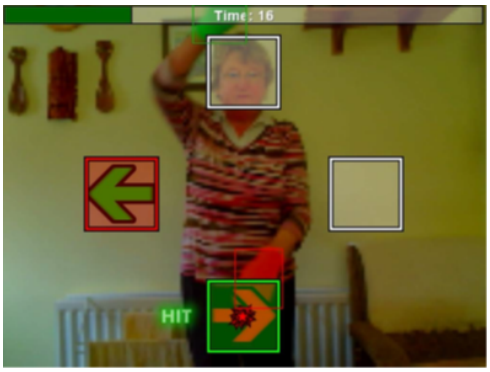


Figura 2.5: Exemplo do jogo *Arrow Attack*. Fonte: Burke *et al.* [25].

Bubble Trouble - Neste jogo, aparecem bolhas no ecrã em posições aleatórias que se movem também aleatoriamente, durante um curto intervalo de tempo. Findo este tempo, as bolhas desaparecem. O jogador deverá, por isso, tocar nessas bolhas antes que elas desapareçam, fazendo-as rebentar (ver Figura 2.6). Na versão com os dois braços, cada bolha apresenta a cor de um dos marcadores, sendo que o utilizador deverá rebentar as bolhas com a mão correcta.

Neste estudo efectuado por Marin *et al.* [1], que abrange os exemplos vistos anteriormente, revela-se que, de um modo geral, o uso de jogos como ferramenta de apoio na reabilitação e no treino mostra

Figura 2.6: Exemplo do jogo *Bubble Trouble*, versão com as duas mãos. Fonte: Burke *et al.* [25].

um impacto positivo nos idosos. Todavia, alguns estudos também notaram que houve idosos que interpretaram esta prática como uma experiência pouco agradável ou mesmo desagradável, quando os jogos não tiveram em consideração as limitações típicas desta faixa etária. Marin *et al.* [1] também apontaram a existência de poucos estudos que liguem os idosos aos *serious games*, com o propósito de melhorar a sua saúde.

2.2 *Robot-Assisted Training*

Avanços na robótica tornaram possível a aplicação de robots como meios de assistência para o Homem, podendo ainda ser específicos para um determinado público-alvo, desde crianças, idosos, estudantes, a pacientes que sofreram um AVC. *Robot-Assisted Training* (RAT) é uma área de investigação em crescimento no domínio da Interacção Humano-Robot. Esta área estuda a forma como os robots podem interagir com os humanos, auxiliando-os na realização de determinadas tarefas de treino físico e cognitivo [26]. Estes sistemas podem ser aplicados nas mais diversas áreas desde a reabilitação pós-AVC, à terapia com crianças com Perturbações do Espectro do Autismo (PEA), ao treino cognitivo para doentes com demência ou doença de Alzheimer, entre outras aplicações.

Tsiakas *et al.* [26] propuseram um conjunto de categorias taxonómicas para caracterizar os sistemas de *Robot-Assisted Training*. Esta classificação divide-se em tipo de tarefa e os seus requisitos, tipos de interacção e as suas funções, níveis de autonomia e de aprendizagem e dimensões de personalização, tal como se pode observar na Figura 2.7. De seguida, explica-se com mais pormenor cada uma destas categorias em conjunto com a apresentação de exemplos de sistemas RAT. Na Tabela 2.4 encontram-se sintetizados todos estes exemplos.

O tipo de tarefa e os seus requisitos são os primeiros parâmetros a serem definidos quando se projecta um sistema desta natureza, uma vez que são o ponto de partida para a definição dos restantes e para a criação do sistema em si. Este conjunto de parâmetros estabelece a finalidade do sistema (exemplo: reabilitação, educação, entre outras), a morfologia do robot, o nível de criticidade das tarefas, o público-alvo, os canais *input-output* necessários e o tipo de assistência proporcionada por este sistema, tal como está apresentado na Figura 2.7. Segundo Tsiakas *et al.* [26], os tipos de assistência podem ser *Physically Assistive Robotics* (PAR), *Socially Assistive Robotics* (SAR) e *sensory*

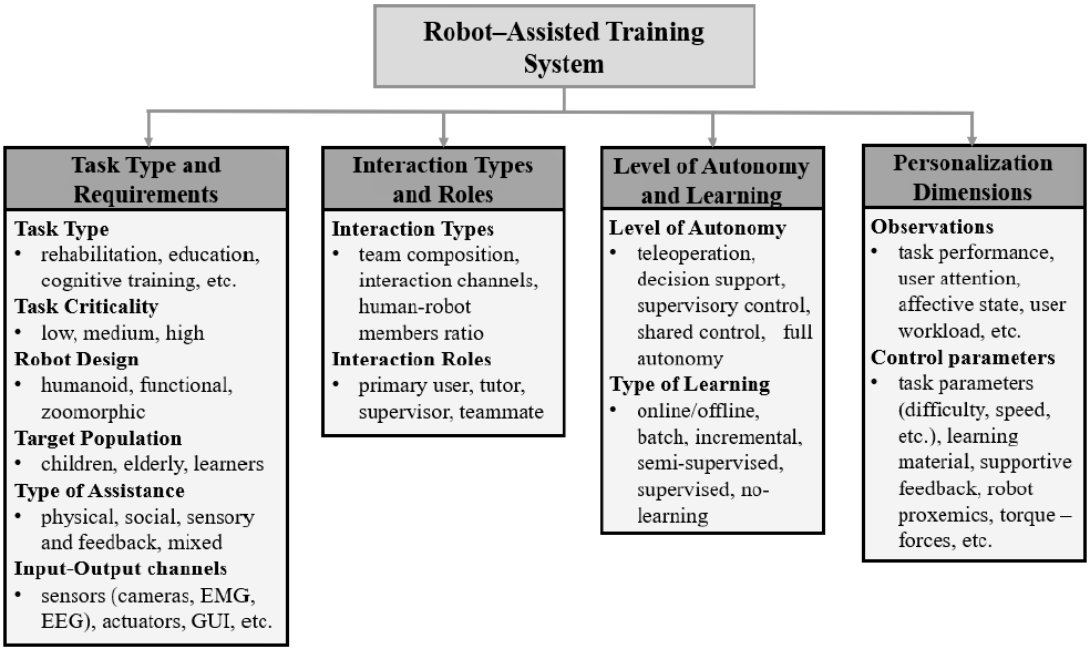


Figura 2.7: Categorias taxonómicas para o *Robot-Assisted Training*. Fonte: Tsiakas *et al.* [26].

Tabela 2.4: Classificação de sistemas de *Robot-Assisted Training*. Adaptado de Tsiakas *et al.* [26].

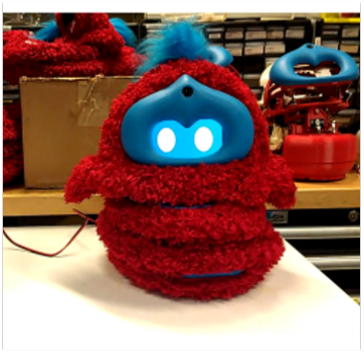
Tipo de Tarefa e Requisitos	**Tipos de Interacção e Funções**	**Níveis de Autonomia e de Aprendizagem**	**Dimensões de Personalização**
Adaptative Upper-Limb Rehabilitation using a Haptic device	O braço robótico treina o utilizador numa tarefa de alcance dos membros superiores. As decisões do sistema são monitorizadas por um supervisor.	O robot actua autonomamente com base numa determinada *policy* (não há aprendizagem). Um especialista pode alterar essa acção.	O sistema decide atingir um certo objectivo, com um determinado nível de resistência e quando a tarefa deve terminar.
Socially Assistive Robotics for Language Learning with Children	O robot social actua como um tutor durante um jogo de aprendizagem de idiomas.	O robot age de forma totalmente autónoma e aprende através de *Reinforcement Learning*.	O robot ajusta o seu envolvimento e valência durante as instruções verbais.
SAR-based system for Post Stroke Rehabilitation for Elderly Patients	O robot monotoriza, incentiva e auxilia os seus utilizadores durante a reabilitação.	O robot actua de forma totalmente autónoma e personaliza a sua *policy* usando o *Policy Gradient Reinforcement Learning*.	A exigência da reabilitação é ajustada de acordo com o desempenho do utilizador.
Robot-Based Rehabilitation using Serious games and Haptic device	O utilizador executa uma tarefa de alcance usando um dispositivo háptico.	O robot age de forma autónoma e aprende através de *Reinforcement Learning*.	O sistema ajusta os parâmetros do jogo de modo a desafiar o seu utilizador.

and feedback systems.

Physically Assistive Robotics corresponde à área que estuda como é que os robots podem dar assistência aos seus utilizadores através da interacção física. Por exemplo, o *Adaptative Upper-Limb Rehabilitation* [27], apresentado na Tabela 2.4, consiste num sistema que presta assistência física a pacientes que sofreram de AVCs. Este sistema guia os seus utilizadores numa tarefa de reabilitação dos membros superiores e possui um mecanismo de tomada de decisão que permite adaptar automaticamente os parâmetros do exercício às capacidades dos seus utilizadores [27]. Na Figura 2.8(a) está representado o dispositivo de reabilitação robótico utilizado neste exercício.

No âmbito da *Socially Assistive Robotics*, apresenta-se o exemplo *Socially Assistive Robotics for Language Learning with Children*, também referido na Tabela 2.4. Gordon *et al.* [28] desenvolveram este sistema que permite às crianças aprenderem uma segunda língua na companhia de um robot totalmente autónomo, de nome Tega (ver Figura 2.8(b)). O sistema usa uma câmara para captar e analisar as expressões faciais, como o olhar e o sorriso, e as características afectivas, com o objectivo de fornecer uma resposta personalizada, através do seu comportamento social verbal. Este robot foi desenvolvido para estabelecer uma interacção a longo prazo com as crianças [29].

(a) Dispositivo robótico usado na reabilitação pós-AVC.



(b) Robot Tega utilizado na aprendizagem das crianças.

Figura 2.8: Exemplos de sistemas robóticos. Fontes: (a) Kan *et al.* [27] e (b) Westlund *et al.* [29].

Relativamente aos sistemas sensoriais e de *feedback*, o seu objectivo é a recolha de informações, não só do estado do robot como também do utilizador, através dos canais de entrada. Estas informações do utilizador tanto podem ser referentes ao seu desempenho como ao seu estado afectivo. O comportamento do robot é, posteriormente, gerado com base no processamento das informações de entrada e é expresso através dos canais de saída associados, por exemplo, ao movimento do robot, ao desenvolvimento de estados emocionais ou ao ajuste de parâmetros da tarefa.

Voltando às restantes categorias taxonómicas, é necessário estabelecer como é feita a interacção entre o robot e o Homem. A interacção mais simples consiste na comunicação entre um robot e um único utilizador. Num caso mais complexo, podem existir vários utilizadores ou até diferentes níveis de utilizadores. As diferentes categorias de interacção num sistema RAT são utilizador primário, treinador/tutor, supervisor e colega de equipa (no mesmo nível do utilizador primário). Considera-se um utilizador primário aquele que participa activamente na tarefa, por exemplo, um aluno ou um paciente. Apesar de se assumir que o utilizador primário é sempre um humano, existem trabalhos que se focam em treinar um utilizador secundário (por exemplo, um terapeuta) e, neste caso, o robot assume o papel de utilizador primário. Nos exemplos mencionados na Tabela 2.4, o robot assume quase sempre o papel de tutor.

O nível de autonomia do robot também constitui um factor importante, uma vez que o robot tanto pode ser totalmente autónomo, como pode estar dependente, em maior ou menor proporção, da orientação ou controlo de um humano. A autonomia do robot divide-se em duas vertentes, a da percepção e a do controlo comportamental. A autonomia da percepção diz respeito ao quanto o robot consegue perceber o ambiente que o rodeia, com ou sem a intervenção humana. O módulo da

percepção é responsável por recolher e analisar as informações recebidas através dos sensores disponíveis, com a finalidade de entender as intenções dos utilizadores, criar um modelo do comportamento humano e detectar outras situações durante a tarefa, como por exemplo, as expressões faciais. A autonomia do controlo comportamental refere-se à quantidade de intervenção humana necessária na tomada de decisões, na execução de acções e no seu planeamento. Este módulo usa as informações obtidas anteriormente para seleccionar e executar uma determinada resposta, com o objectivo de ajudar o utilizador durante a tarefa. No exemplo da *Adaptative Upper-Limb Rehabilitation*, já referido anteriormente, o sistema sugere uma acção ao supervisor, sendo que este concorda ou discorda da decisão do sistema, e toma a decisão final [27]. Neste caso, o sistema robótico é autónomo na percepção, mas necessita do controlo de um supervisor na execução de respostas.

Em relação à aprendizagem, os robots também podem aprender ou não com a sua experiência e com o ambiente envolvente. Existem diferentes tipos de aprendizagem nos robots, podendo esta ser baseada em modelos, supervisionada ou não-supervisionada, *online* ou *offline*, entre outras. Várias abordagens de aprendizagem podem ser aplicadas a sistemas robóticos, tais como *Machine Learning*, *Active Learning* e *Reinforcement Learning*, conforme está indicado na Figura 2.7. À semelhança dos níveis de autonomia, a aprendizagem também se pode aplicar tanto na percepção, como no controlo comportamental. No sistema *Robot-based rehabilitation using serious games and haptic device* [30], referido na Tabela 2.4, é utilizado o método de *Reinforcement Learning* para aprender um modelo de como o utilizador joga a fim de ajustar a dificuldade do jogo, em tempo real.

Por fim, as dimensões de personalização pretendem adaptar o melhor possível os parâmetros e o comportamento do robot ao desempenho do utilizador. Tsiakas *et al.* [26] referem-se às dimensões de personalização como um conjunto de observações feitas pelo robot a fim de ajustar os seus parâmetros e, consequentemente, o seu comportamento. Um dos objectivos da pesquisa sobre a personalização recai na forma como os parâmetros observados podem ser utilizados para ajustar os parâmetros de controlo à interacção do robot. Quer isto dizer, se um robot conseguir analisar os sinais emocionais e sociais provenientes dos seus utilizadores, torna mais fácil a personalização da interacção humano-robot, em tempo real. No exemplo *Socially Assistive Robotics for Language Learning with Children* [28], o sistema tenta estimar o estado emocional da criança e ajustar o seu comportamento conforme esses dados, seleccionando estratégias motivacionais apropriadas.

2.3 *Virtual coaches*

Os *virtual coaches* também emergiram com a evolução da tecnologia. A sua missão consiste na monitorização contínua das actividades dos seus utilizadores e do seu ambiente envolvente, no sentido de detectar situações em que seja necessário intervir, dando assistência imediata [31]. Uma das aplicações mais importantes destes dispositivos é o auxílio de pessoas com capacidade cognitiva reduzida, não só pelo envelhecimento natural, como por doenças mentais ou lesões traumáticas. O declínio na memória e na capacidade cognitiva acaba por afectar não só as próprias pessoas como os seus cuidadores e familiares. Além disso, muitas vezes quando os pacientes saem do hospital não têm o

treino necessário para continuar a ter os cuidados de saúde recebidos até então. Por exemplo, no que diz respeito ao manuseamento de dispositivos médicos recém-prescritos ou no seguimento de regimes médicos mais complexos. O não cumprimento adequado das instruções médicas pode levar a um novo internamento. Os *virtual coaches* podem actuar neste sentido, garantindo o cumprimento das instruções de reabilitação e o seguimento dos novos regimes médicos.

À medida que o utilizador progride nas suas capacidades, o dispositivo adapta-se às suas novas competências e pode reduzir o número de instruções e lembretes dados. Por outro lado, caso o utilizador necessite de mais instruções, o *virtual coach* assim o fará, no sentido de fornecer um maior suporte. Um prestador de cuidados de saúde também pode introduzir novas instruções no dispositivo, sem ter que haver uma visita entre ambas as partes. Além disso, o *virtual coach* através da sua observação constante do paciente, pode transmitir mais informações ao seu médico, complementado os seus exames periódicos.

Siewiorek *et al.* [31] revelam cinco partes essenciais na arquitectura de um *virtual coach*. Estas encontram-se representadas na Figura 2.9, assim como a relação existente entre elas. Estes dispositivos monitorizam o desempenho do utilizador e o seu contexto (*Sensor processing*), definem qual é a resposta mais apropriada de acordo com o ambiente (*Coaching model*) e transmitem o seu *feedback* e incentivo ao utilizador (*User engagement*). Tal como já foi referido, um prestador de cuidados de saúde pode introduzir novos dados no sistema se necessário (*Prescription*) e à medida que o utilizador progride, o *feedback* dado pelo *virtual coach* também evolui (*User interaction*).

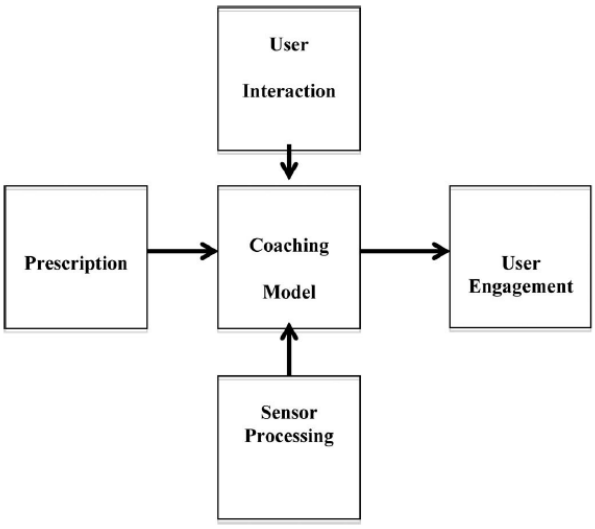


Figura 2.9: Representação dos cinco elementos de um *virtual coach*. Fonte: Siewiorek *et al.* [31].

De seguida são apresentados três exemplos de *virtual coaches*. O *Manual Wheel Chair Propulsion Coach* supervisiona o movimento dos braços quando se impulsiona uma cadeira de rodas, o IMPACT pretende motivar os seus utilizadores a serem fisicamente activos e o *Mem-Exerciser* auxilia a memória dos seus pacientes. A Tabela 2.5 apresenta uma síntese destes três exemplos, descritos nas cinco partes essenciais da arquitectura de um *virtual coach*.

Tabela 2.5: Exemplos de *virtual coaches*. Adaptado de Siewiorek *et al.* [31].

Sistema	**Função**	***Prescription***	***Sensor Processing***	***User Interaction***	***Coaching Model***	***User Engagement***
Manual Wheel Chair Propulsion Coach	Corrigir os movimentos	Treino supervisionado	Acelerómetro	Movimento de propulsão	Aprendizagem automática	Áudio
IMPACT	Motivar os utilizadores	Instruções escritas	Pedómetro e GPS	Diário	*Rule-based*	Gráfico
MemExerciser	Recordar experiências	Amostragem automática	Câmara, GPS e microfone	Reproduzir o conteúdo	Cuidador	Imagens e Áudio

O *virtual coach* do exemplo da *Manual Wheel Chair Propulsion*, representada na Figura 2.10, pretende aconselhar os seus utilizadores a evitarem formas de locomoção perigosas. Neste estudo são usados acelerómetros nos pulsos e na cadeira de rodas, de modo a analisar a forma como os utilizadores realizam os movimentos. A principal medida analisada neste sistema é o padrão de propulsão do utilizador. Siewiorek *et al.* [31] identificaram quatro padrões de propulsão que os utilizadores tendem a seguir, sendo estes: semicircular, *loop* único, *loop* duplo e arco. Este sistema usa algoritmos de *Machine Learning* para classificar os padrões de propulsão juntamente com o material das superfícies onde se movem. Observou-se que quanto maior a resistência da superfície atravessada, maior a precisão da previsão de propulsão. Quando não é o próprio utilizador a controlar a cadeira, o *feedback* é suprimido.

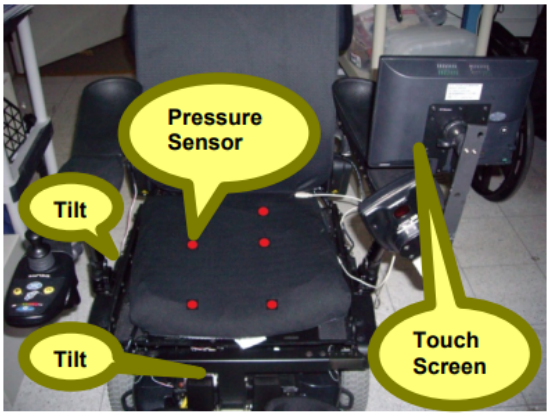


Figura 2.10: Cadeira de rodas com o *virtual coach* indicado. Fonte: Siewiorek *et al.* [31].

O sistema *IMPACT* contabiliza o número de passos dados pelo utilizador, juntamente com a sua localização, através de um telemóvel com GPS. O aplicativo ainda solicita ao utilizador que introduza informações sobre os sítios onde se deslocou e com quem esteve, de modo a tornar a actividade mais personalizada. Com isto, pretende-se aumentar a motivação dos seus utilizadores. Este sistema guarda também um histórico com todas as actividades. Num estudo conduzido por Siewiorek *et al.* [31], verificou-se que os utilizadores ganharam mais consciência relativamente à importância da actividade física.

O *MemExerciser* dirige-se a pessoas que estejam num estado inicial da doença de Alzheimer. Nestas situações, as pessoas mantém as memórias antigas mas deixam de se lembrar de episódios mais

recentes, o que pode gerar sentimentos de frustração, raiva e depressão. Além de condicionar as próprias pessoas, os seus familiares e cuidadores também são afectados. A forma como os cuidadores costumam proceder é dando pistas ou detalhes sobre a experiência para que a pessoa se consiga lembrar. O sistema *MemExerciser* pretende actuar neste sentido, resolvendo esta lacuna. Este sistema utiliza uma abordagem de captura passiva para que o utilizador não precise de se lembrar de iniciar a gravação. As várias experiências são gravadas através de uma câmara, de um gravador de áudio e de um GPS [31]. Numa primeira instância, é feita uma análise automática pelo computador. Posteriormente, o cuidador verifica as sugestões dadas pelo sistema para a construção da narrativa e selecciona as imagens e pistas mais relevantes para cada uma das experiências vividas, juntamente com anotações visuais ou em formato áudio. Por fim, o sistema estrutura toda a informação escolhida para estimular a memória do utilizador. Na Figura 2.11, está representada a interface mostrada ao utilizador.

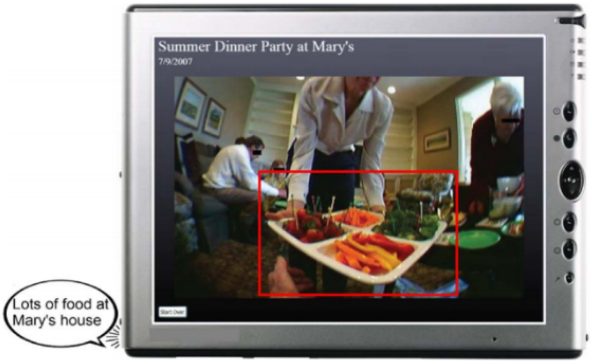


Figura 2.11: Interface do *MemExerciser* apresentada ao utilizador. Fonte: Siewiorek *et al.* [31].

2.4 Exemplos

2.4.1 *Interative Exercise Coaching System*

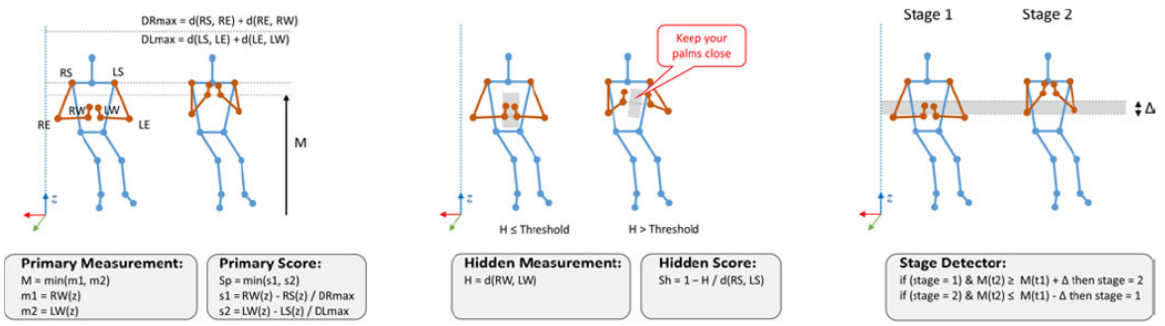
Ofli *et al.* [32] referem que além da nutrição e da socialização na população mais idosa, os exercícios físicos e cognitivos estão entre as abordagens preventivas mais eficazes que melhoram a sua qualidade de vida. Ainda assim, muitos idosos continuam a enfrentar dificuldades que os afastam destas práticas, não só por razões sociais e económicas, como também por falta de motivação e de confiança.

Com a chegada de novas tecnologias que permitem a detecção dos movimentos dos utilizadores, tornou-se possível elaborar sistemas que permitem o treino de exercícios físicos e cognitivos nas suas próprias casas. Juntamente com a introdução dos *exergames* no mercado, que combinam estes exercícios com videojogos, pretende-se aumentar a motivação da população e promover, simultaneamente, um estilo de vida mais activo nas várias faixas etárias.

Apesar de já terem sido lançadas várias aplicações de *fitness* ao longo do tempo, muitas destas não foram projectadas para os idosos, mas sim para a população mais jovem. Por conseguinte, estas não são adequadas às limitações e às dificuldades características dos mais velhos.

Neste sentido, Ofli *et al.* [32] desenvolveram um sistema de treino que conduz os seus utilizadores num conjunto de exercícios em vídeo, rastreando e medindo os seus movimentos, ao mesmo tempo que proporciona *feedback* em tempo real do desempenho do utilizador e o grava ao longo do tempo. Este sistema pretende melhorar a flexibilidade, o equilíbrio, a força e a resistência dos idosos, de modo a reduzir o risco de quedas e promover a sua independência nas actividades diárias. Neste sistema, utiliza-se uma câmara *Kinect* que permite a reconstrução esquelética do utilizador, a partir de 20 articulações. Assim, é possível ter uma estimativa da sua postura corporal em tempo real, sendo esta utilizada para avaliar o desempenho do jogador, para realizar a contagem das repetições bem executadas e para enviar alertas de *feedback* nos momentos necessários. Cada exercício inicia-se com um vídeo de um treinador a indicar quais são os seus benefícios para a saúde e a fazer uma demonstração dos movimentos necessários para essa tarefa. Após este passo inicial, os jogadores começam o seu exercício, por conta própria, seguindo o *feedback* que é dado ao longo do jogo. No fim, o desempenho do utilizador é registado e é-lhe resumido, havendo também um histórico que permite visualizar o resumo mensal ou das últimas 10 sessões desse jogador.

Ofli *et al.* [32] estudaram 12 exercícios, de entre 130 sugeridos por Sue Scott no livro *Able Bodies Balance Training* [33], que visam melhorar o equilíbrio e a confiança nos idosos e reduzir o risco de quedas. Um deles é o exercício *Buddha's Prayer*, que consiste em elevar e baixar os braços, sempre com as palmas das mãos juntas. Este exercício é para ser realizado sentado e o seu objectivo é levantar as mãos o máximo que se conseguir, sem nunca separar as palmas das mãos. As articulações relevantes neste exercício estão assinaladas a cor laranja na Figura 2.12.



(a) Avaliação da *performance*. (b) Verificação do movimento. (c) Detecção do estado do jogo

Figura 2.12: Análise do exercício *Buddha's Prayer*. Fonte: Ofli *et al.* [32].

Na Figura 2.12(a), é possível observar a medida primária M, que corresponde à altura mínima dos pulsos, dada por $M = min(RW(z), LW(z))$, sendo $RW(z)$ a altura do pulso direito e $LW(z)$ a altura do pulso esquerdo. Este mínimo serve para assegurar que existe uma elevação suficiente das mãos. A pontuação primária calcula-se através do mínimo da razão entre a diferença de alturas do pulso e do ombro com o comprimento do braço, para cada um dos lados (direito e esquerdo). A normalização feita pelo tamanho do braço corresponde à altura máxima de elevação das mãos. Esta pontuação pode ser observada nas *performance bars*, representadas no ecrã de *feedback*. Na Figura 2.12(b) é feita a verificação dos movimentos. A *hidden measurement* H corresponde à distância entre os pulsos. Se essa distância ultrapassar um certo limite, *threshold*, o sistema envia uma mensagem a alertar o

utilizador. Além disso, o sistema também requer que o utilizador esteja numa posição erecta e, por isso, também são analisados os ângulos de inclinação dos membros superiores do utilizador em relação ao chão. Na Figura 2.12(c) é revelado o método de contagem das repetições do exercício, através da detecção de dois estados. Fixando um intervalo Δ, se a medida primária M ultrapassar esse valor, o estado muda do 1 para o 2. Por outro lado, partindo do estado 2 e se essa mesma medida diminuir além desse intervalo Δ, altera-se para o estado 1.

A Figura 2.13 apresenta um exemplo ilustrativo de como o *feedback* é transmitido ao utilizador. É possível observar que na parte superior do ecrã se encontra o número do exercício, o seu nome e o número de repetições conseguidas com sucesso. As *performance bars* ilustram o desempenho do utilizador sendo que o seu comprimento avalia a pose actual do utilizador, ou seja, as barras são dinâmicas e estão constantemente a oscilar de acordo com a postura do jogador. No ecrã também vão aparecendo mensagens e alertas de *feedback*, tais como "sit tall" e "keep your feet on the ground", como é visível na Figura 2.13. Estas mensagens também são transmitidas via áudio. Adicionalmente, são emitidos sons curtos (exemplo: "ding") sempre que a repetição de movimentos é efectuada com sucesso.

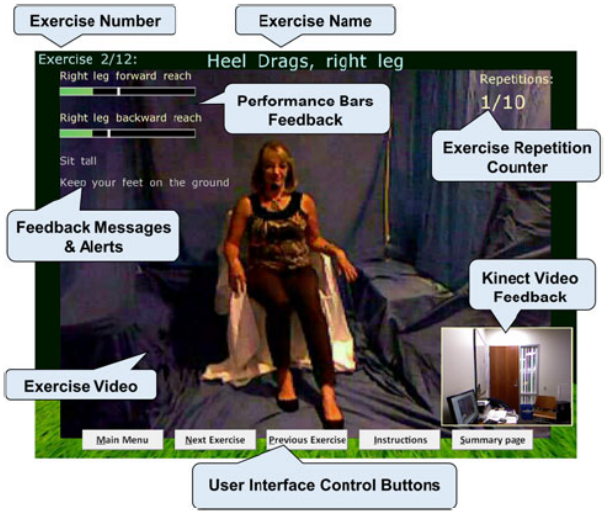


Figura 2.13: Ecrã de *feedback* apresentado durante os jogos. Fonte: Ofli *et al.* [32].

Neste estudo participaram 6 idosos, com uma média de idades de 81 anos. Todos eles consideraram o ecrã de *feedback* útil durante o exercício e gostaram de receber a motivação áudio ao longo das tarefas. Também consideraram as demonstrações iniciais úteis para o desempenho correcto dos exercícios. Contudo, alguns questionaram se o sistema conseguia mesmo observá-los e outros relataram a dificuldade em perceber como é que as *performance bars* se relacionavam com os seus movimentos. As mensagens de *feedback* textuais foram consideradas demasiado pequenas para serem lidas a partir de uma certa distância.

De um modo geral, os utilizadores apreciaram os exercícios e o facto de os poderem realizar em casa. Embora este estudo tenha tido algumas limitações no número de participantes e na duração dos testes, foi possível confirmar o potencial destes sistemas de exercícios interactivos na promoção da actividade física nos idosos.

2.4.2 *Adaptative Mixed Reality Rehabilitation*

A *Adaptative Mixed Reality Rehabilitation* (AMRR) corresponde a um novo sistema desenvolvido para ajudar os pacientes durante um exercício de reabilitação [34]. Este sistema promove o treino das componentes motoras, avaliando a qualidade dos movimentos executados e fornecendo *feedback* para guiar os seus utilizadores.

Portanto, a AMRR consiste num treino repetitivo de exercícios, com o auxílio de objectos específicos, onde o movimento executado é analisado em tempo-real, sendo fornecidas medições cinemáticas precisas. Estes dados cinemáticos também são utilizados para produzir ao utilizador um *feedback* em tempo real, para que este possa auto-avaliar a qualidade dos seus movimentos. As interacções com os participantes incentivam a conclusão dos exercícios e a aprendizagem das componentes motoras implicadas. Por este motivo, as tarefas e os *feedbacks* podem ser direccionados tanto para a conclusão dos exercícios como para a execução dos mesmos, dependendo das necessidades terapêuticas de cada paciente.

Num estudo realizado por Duff *et al.* [34] foram comparados os resultados da AMRR com a reabilitação tradicional dos membros superiores, em pacientes que sofreram de ataques cardíacos. Esta terapia foca-se na combinação de exercícios que incidem em várias deficiências e na motivação do paciente para realizar as tarefas repetitivas, promovendo a sua aprendizagem.

O *feedback* é transmitido aos participantes em formato visual e de áudio. No *feedback* visual mostra-se o erro da trajectória e da rotação da mão e no *feedback* auditivo são dadas indicações sobre a velocidade do movimento da mão, a extensão do cotovelo e a compensação do tronco e do ombro. Ambos os *feedbacks* são fornecidos em tempo real e o seu principal objectivo é avaliar o desempenho do paciente, adaptando-se com o desenrolar da actividade. No final também é dado um sumário sobre a prestação do paciente e os seus principais erros, no sentido de os melhorar no futuro. Na Figura 2.14 está representado um paciente a usar o sistema AMRR.

Este estudo revelou que os pacientes que receberam a terapia AMRR mostraram uma melhoria consistente nas medidas cinemáticas, levando Duff *et al.* [34] a concluir que esta terapia demonstra ser uma maneira eficaz de melhorar a qualidade dos movimentos dos seus utilizadores. Todavia, também referem ser necessário mais estudos para comprovar os benefícios desta terapia a longo prazo.

Figura 2.14: Paciente em terapia com a AMRR. Fonte: Duff *et al.* [34].

Capítulo 3

Definição do estudo e metodologias

Para a realização deste trabalho, utilizou-se a plataforma portátil de *exergames* para idosos, PEPE, apresentada na Figura 3.1. Este sistema consiste numa plataforma de jogos de realidade aumentada e pretende promover a prática desportiva nos idosos, contribuindo para a sua autonomia nas tarefas do dia-a-dia e proporcionando, simultaneamente, uma actividade de entretenimento [35]. Os *exergames* são projectados no chão e através de um sensor *Kinect* são captados os movimentos dos idosos, permitindo desta forma controlar os elementos do jogo [36]. A *Kinect* realiza medições de 25 articulações do corpo humano, identificadas na Figura 3.2. Por esta razão, não é necessário que o jogador possua algum sensor ou dispositivo para a captação dos seus movimentos, o que poderia ser um obstáculo para os mais idosos.

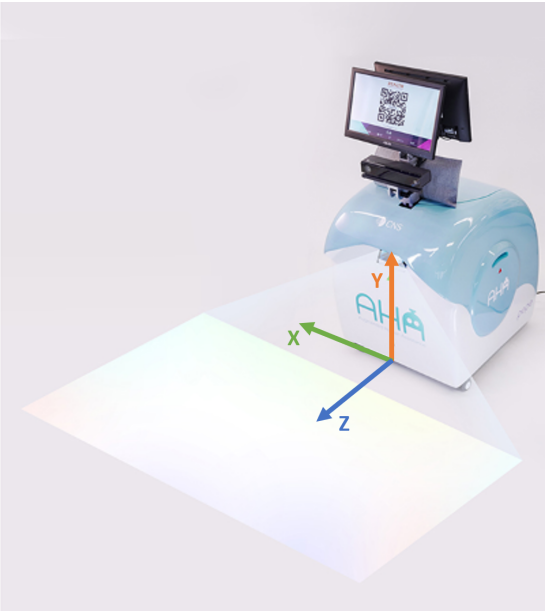


Figura 3.1: Sistema robótico PEPE, com a representação do referencial, a partir do qual são realizadas as medições da *Kinect*. Adaptado de Augmented Human Assistance [35].

3.1 *Exergames*

Os *exergames* considerados neste trabalho foram desenvolvidos tendo em consideração os idosos como público-alvo e procuram treinar a sua capacidade motora (equilíbrio, agilidade e flexibilidade), a resistência aeróbica e a força muscular. A tradição portuguesa serviu de inspiração para a criação destes jogos, que são apresentados com maior detalhe nos pontos que se seguem [37, 38].

Grape Stomping - Corresponde a um jogo de resistência aeróbica cujo objectivo é pisar os cachos de uvas, que vão aparecendo na área de jogo, transformando-os em vinho. Os joelhos devem ser alternadamente elevados até uma altura ajustável, num movimento de marcha estática. O jogo permite ser realizado em pé ou sentado, de forma a ser mais abrangente, e pode também exigir o movimento de flexão e extensão dos braços.

Rabelos - O objectivo deste jogo é remar um barco Rabelo e recolher os barris de vinho que se encontram nas margens do rio, desviando-se simultaneamente das rochas. Este jogo exercita os membros superiores dos participantes e exige uma rotação lateral do tronco. Também pode ser realizado em pé ou sentado, sendo que no último caso é apenas necessário remar.

Toboggan Ride - Este jogo foca-se no treino do equilíbrio e da força muscular do tronco. O jogador deve guiar um carrinho de vime, característico da ilha da Madeira, e recolher os cachos de bananas que vão aparecendo na estrada. O tronco controla o carrinho, ou seja, se houver uma inclinação do tronco para a frente, o carrinho acelera, se houver uma inclinação para a direita, o carrinho desloca-se para a direita, etc. É também necessário evitar os obstáculos que vão surgindo no percurso e, tal como nos jogos anteriores, também pode ser jogado de pé ou sentado.

ExerFado - Neste jogo pretende-se treinar a força muscular e a agilidade dos membros inferiores. É apresentado ao jogador um piano, onde notas musicais se movem ao longo de uma tecla. O utilizador deve posicionar-se em frente à tecla de onde está a vir a nota musical. A sua inspiração partiu das tradicionais casas de Fado de Lisboa e podem ser feitas várias personalizações, tais como a escolha da música, a velocidade a que caem as notas musicais e a quantidade de notas que aparecem.

ExerPong - Inspirado no jogo clássico Pong, os jogadores controlam uma barra lateralmente, seja com um braço, no caso de se jogar sentado, ou com a cintura, se se jogar de pé. O objectivo é destruir o padrão de tijolos com uma bola e, ao mesmo tempo, impedir que a mesma saia do ecrã de jogo, com o auxílio da barra controlável. O tamanho desta vai diminuindo à medida que acerta na bola, de modo a ajustar a dificuldade da tarefa. Em sentido contrário, quando o utilizador deixa escapar a bola, a barra aumenta o seu tamanho, de forma gradual.

3.2 Dados dos jogos

Os dados utilizados neste trabalho foram recolhidos em 5 visitas à Residência Sénior de Belverde, no Seixal. Contribuíram para este estudo 22 idosos, com idades superiores a 70 anos, e 5 fisioterapeutas. Os idosos foram seleccionados pelos fisioterapeutas, de acordo com as suas capacidades mentais, uma vez que deveriam conseguir perceber o jogo e o seu objectivo. Em termos de capacidades físicas, não houve muitas restrições pois cada jogo dispunha da opção de se realizar sentado. No total, foram realizados 158 jogos: 34 do jogo Toboggan Ride, 39 do Grape Stomping, 34 do ExerPong, 40 do Rabelos e 11 do ExerFado.

O sistema realiza 60 leituras por segundo, recolhendo dados do estado actual do jogo e da posição do jogador. Tal como seria de esperar, os parâmetros que caracterizam o estado actual do jogo variam consoante o seu objectivo, sendo que cada jogo tem o seu conjunto de parâmetros. Por exemplo, o jogo Toboggan Ride verifica, em cada leitura, o número de cachos de bananas capturados até ao momento. Por outro lado, no jogo ExerPong são contabilizadas as bolas perdidas até esse instante.

Relativamente à posição do jogador, em cada leitura são efectuados registos das 25 articulações, detectadas pela *Kinect V2*. A identificação das articulações está representada na Figura 3.2 e estas medições são iguais para os 5 *exergames*.

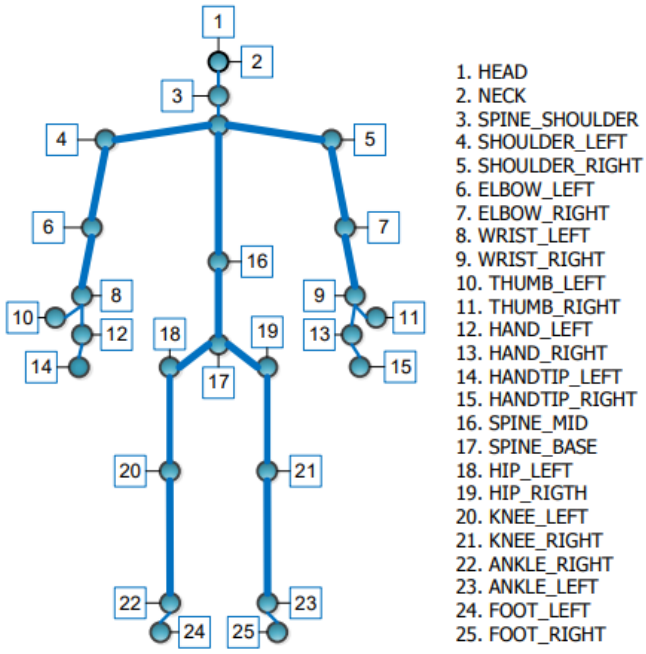


Figura 3.2: Articulações identificadas pela *Kinect V2*. Fonte: Ahmed *et al.* [39].

O referencial a partir do qual são feitas as medições das articulações está representado na Figura 3.1. A sua origem encontra-se no ponto do plano do chão, imediatamente por baixo do sensor *Kinect*. Os eixos XZ definem o plano do chão, os eixos YZ definem o plano vertical do centro da câmara e os eixos XY constituem o plano da câmara. É de notar ainda que cada articulação tem o seu referencial. Porém, estes não são apresentados uma vez que não são relevantes para este trabalho.

Portanto, para cada articulação, registam-se os seguintes dados:

JointType - nome da articulação, que permite a sua identificação;

PositionX, *PositionY*, *PositionZ* - coordenadas XYZ da articulação em metros, sendo que Y corresponde à altura desta relativamente ao plano do chão, X refere-se ao desvio lateral relativamente ao plano vertical do centro da câmara e Z indica a distância da articulação ao plano da câmara;

OrientationX, *OrientationY*, *OrientationZ*, *OrientationW* - rotação do referencial da articulação relativamente ao referencial indicado na Figura 3.1, em quaterniões.

3.3 *Feedback*

A execução dos vários jogos foi filmada para que fosse possível guardar a *performance* do jogador, juntamente com o *feedback* verbal dado pelos fisioterapeutas no decorrer dos mesmos. Desta forma, obtiveram-se no total 158 gravações de vídeo, que mostram a projecção do jogo no chão e o idoso, garantindo a protecção da sua identidade. Foi necessário efectuar a transcrição desses 158 jogos, de modo a obter, por escrito, todas as informações transmitidas aos idosos durante os mesmos.

Com base na análise dos estudos de Johnson *et al.* [4] e Burgers *et al.* [8], o *feedback* foi dividido em 3 vertentes: explicativo, correctivo e motivacional, tendo-se estabelecido uma definição para cada um deles. O *feedback* explicativo dá informações ao jogador de como este deve proceder para atingir a resposta correcta, fornece pistas que permitam melhorar o seu desempenho e explica porque é que uma determinada resposta estava certa ou porque é que estava errada. Por outro lado, o *feedback* correctivo apenas indica se a resposta do jogador, num certo momento do jogo, está correta ou incorrecta, não fornecendo qualquer informação adicional. Por último, o *feedback* motivacional refere-se a mensagens de incentivo, para que o jogador não desista e consiga chegar ao fim do jogo.

Cada uma das transcrições foi etiquetada com estas 3 *labels* distintas. Após uma análise geral das mensagens dos fisioterapeutas, foram estabelecidas certas considerações que permitiram realçar o conceito de cada um dos *feedbacks*. Estas considerações foram adaptadas aos jogos em questão e apresentam-se sintetizadas na Tabela 3.1, juntamente com exemplos representativos, retirados directamente das transcrições.

3.4 Selecção

Como se pretende, com este trabalho, incentivar os idosos a praticar exercício físico, com o recurso aos *exergames*, optou-se por se fazer a replicação dos *feedbacks* pertencentes à vertente motivacional. É de notar que, só por si, o *feedback* já é considerado como uma motivação nestes jogos. Contudo, era necessário dar ênfase às mensagens de carácter motivacional e, por isso, estas foram o foco deste trabalho.

Foi feito o levantamento e contagem de todas as mensagens etiquetadas com a *label* de *feedback* motivacional. Como seria de esperar, existem muitas palavras e expressões que, mesmo diferentes,

Tabela 3.1: Síntese das considerações assumidas para cada tipo de *feedback*.

Tipo de *feedback*	Definição	Exemplos
Explicativo	Quando se explica o que está a acontecer no jogo (percepção geral do jogo).	"Então aqui temos o barco. Temos que fazer o movimento de remar para o barco andar, ok?"
	Quando se dá indicações ou pistas sobre o que fazer, para executar correctamente o exercício.	"Tem a ver com o movimento do braço." "Tem que remar bem, se não o barco pára."
	Quando se justifica o que se está a fazer mal.	"Tem que remar igual com os dois." "Tem que vir atrás buscar a água e ir à frente."
Correctivo	Quando o jogador está a executar mal o exercício, e é corrigido.	"Não se vire mais para lá, para a frente." "Mais à frente."
	Quando o jogador executa a tarefa correctamente, depois de uma correcção.	"Isso!" "Isso mesmo!"
Motivacional	Quando o jogador está a realizar correctamente o exercício, depois de um período em que está a fazer tudo bem.	"Continue a remar, dona Isabel." "Continue mais um bocadinho. Vá você consegue."
	Quando o jogador está a agir correctamente, e se explica novamente o que fazer para executar a tarefa de forma correcta.	"Vai à frente e volta, vai à frente e volta." "Movimento circular."
	Qualquer interacção mais pessoal entre o jogador e o fisioterapeuta.	"Vamos passear onde, dona Fina?" "Está cansada? Não desista!" "Bora lá!"

têm o mesmo significado. Assim sendo, algumas dessas mensagens foram agrupadas, de acordo com a sua semântica.

No Anexo A estão apresentadas todas as mensagens consideradas de *feedback* motivacional, assim como, o seu número de repetições. É de notar que este número oscila bastante de jogo para jogo, tendo sido destacados, com a cor azul, os conjuntos de *feedbacks* que possuíam mais repetições em cada um dos jogos.

Tendo em conta que cada jogo recolhe diferentes parâmetros, optou-se por estudar apenas um deles. Por conseguinte, foi escolhido o jogo Rabelos por se tratar daquele com o maior número de execuções, tendo sido realizado 40 vezes. Além disso, este jogo é o que contém mais amostras dos *feedbacks* destacados, a azul, na Tabela A.3 no Anexo A, sendo cerca de 886 distribuídas por 7 conjuntos de *feedback* diferentes.

Além dos dados das 25 articulações dos jogadores, também são recolhidos outros parâmetros mais específicos deste jogo, sendo que estes se encontram descritos nos seguintes pontos:

TimeInSeconds - Hora do dia, em segundos;

SessionTimeInSeconds - Tempo desde o início do jogo, em segundos;

PlayerPositionX, *PlayerPositionZ*, *PlayerRotationY* - Coordenadas XZY do barco dentro do jogo. A direcção Z é positiva ao longo do rio, iniciando-se com o valor 0. O eixo X tem a sua origem no centro do rio e assume valores positivos para a direita e negativos para a esquerda e o parâmetro PlayerRotationY é a coordenada Y, em quaternião, referente à rotação do barco;

PickingLeft - Pode tomar o valor 0 ou 1, sendo 1 se o gesto de apanhar os barris para a esquerda foi realizado com sucesso e 0 caso contrário;

PickingRight - Pode tomar o valor 0 ou 1, sendo 1 se o gesto de apanhar os barris para a direita foi realizado com sucesso e 0 caso contrário;

MovingForward - Pode tomar o valor 0 ou 1, sendo 1 se o barco está a avançar nesse *frame* e 0 caso contrário;

Rows - Número de remadas realizadas até ao momento;

Barrels - Número de barris recolhidos até ao momento;

RocksHits - Número de colisões em rochas sofridas até ao momento.

3.5 Metodologia

Neste trabalho, pretende-se implementar no sistema a capacidade de reprodução do *feedback* motivacional dado aos idosos, quando estes estão a jogar os *exergames* referidos anteriormente. Para alcançar este objectivo, é necessário criar um mapeamento que permita ao sistema reproduzir um determinado *feedback* a partir do estado actual do jogo e do jogador. Para isso, é necessário saber previamente as respostas dos fisioterapeutas aos vários estados do jogo. Ao mapeamento referido dá-se o nome de *policy* [40].

Para a aprendizagem de uma *policy* podem ser utilizadas diversas abordagens, entre as quais, a Aprendizagem por Demonstração (em inglês, *Learning from Demonstration*) [40]. Nesta abordagem, uma *policy* desenvolve-se a partir de exemplos, sendo que estes correspondem a sequências de pares estado-acção registadas durante as demonstrações de um professor. Os algoritmos usados na técnica da Aprendizagem por Demonstração utilizam estas sequências para gerarem uma *policy* que reproduza o comportamento desejado. Para clarificar conceitos, as demonstrações do professor dizem respeito às respostas do fisioterapeuta às diferentes situações do jogo, sendo que o professor é um fisioterapeuta.

Argall *et al.* [40] referem que a concepção de uma *policy* é um processo complexo e que, por isso, se encontra restrito aos especialistas do ramo. As abordagens tradicionais que costumam ser utilizadas focam-se em modelos matemáticos que, apesar de teoricamente bem fundamentados, dependem da precisão do modelo do mundo considerado. Além disso, são frequentemente realizadas aproximações, como a linearização, que auxiliam a parte computacional, mas que acabam por prejudicar a precisão do sistema. Outra abordagem para o desenvolvimento de uma *policy* é a Aprendizagem por Reforço (em inglês, *Reinforcement Learning*) [40]. Neste método, a aprendizagem baseia-se na experiência e na exploração dos vários estados. Por cada estado visitado é dada uma recompensa (*reward*) ao sistema, de acordo com a sua conveniência e relevância. Assim, um dos objectivos passa a ser também maximizar essa recompensa. Contudo, e ainda segundo Argall *et al.* [40], definir uma função que forneça recompensas é um processo complexo, pois se o objectivo final não for bem claro, encontrar uma função de recompensa óptima passa por um processo de tentativa-erro ou por uma procura heurística.

Para este trabalho, como se pretende replicar o *feedback* dado nas demonstrações dos fisioterapeutas, optou-se pela Aprendizagem por Demonstração. Esta mostrou vantagens relativamente aos desafios enfrentados pelos outros métodos, pois não é necessário determinar uma função extra que providencie recompensas ao sistema, nem existem as imprecisões das abordagens tradicionais. Deste modo, a *policy* aprende-se a partir de exemplos que combinam o estado do jogo e do jogador com o *feedback* dado pelos fisioterapeutas.

3.5.1 Aprendizagem por Demonstração

Formalmente, a Aprendizagem por Demonstração consiste num conjunto de estados S e acções A, mapeados através de uma função de probabilidade de transição. Um professor reproduz determinadas demonstrações D que o agente deverá conseguir imitar. Geralmente, o estado do mundo S não é completamente observável e, por isso, o agente apenas tem acesso a um estado observável Z, através do mapeamento $M : S \; ! \; Z$. Contudo, neste trabalho, o sistema consegue recolher todas as informações do jogo e do jogador, pelo que o estado do mundo é completamente observável, ou seja, o estado Z corresponde exactamente ao estado S. A *policy* π selecciona a acção a executar, a partir da observação do estado do mundo actual $\pi : Z \; ! \; A$.

Na Figura 3.3 está esquematizado a derivação e a execução da *policy*. Numa primeira fase, esta é derivada a partir das demonstrações de um professor e, posteriormente, selecciona a acção a executar de acordo com o estado actual do mundo.

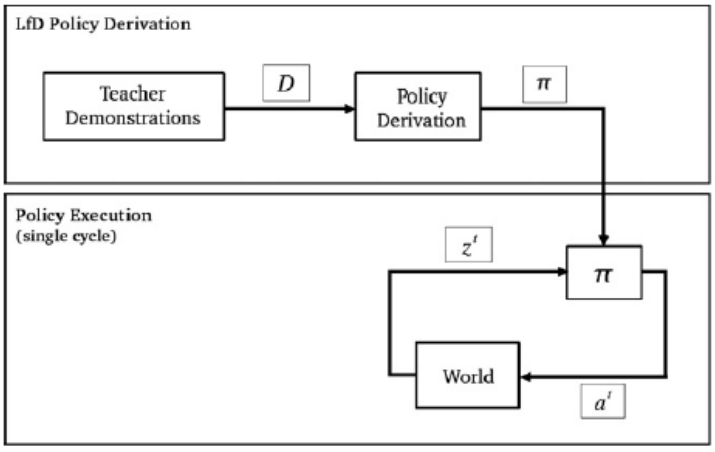


Figura 3.3: Aprendizagem por Demonstração - derivação e execução da *policy*. Fonte: Argall *et al.* [40].

Fazendo o paralelismo com o realizado neste trabalho, a *policy* é aprendida a partir dos exemplos de pares estado-acção (demonstrações), ou seja, para uma determinada situação do jogo e do jogador (estado), o fisioterapeuta proferiu um certo *feedback* (acção). A combinação destes elementos possibilita a criação de um mapeamento (*policy*) que permite ao sistema reproduzir um *feedback* de forma autónoma, olhando apenas para o estado actual do jogo e do jogador (estado actual do mundo).

Existem vários procedimentos para o desenvolvimento de uma *policy* a partir de demonstrações. Na Figura 3.4 encontra-se uma categorização dessas abordagens, que são de seguida explicadas com

maior detalhe. Note-se que o caminho 1-2-3 identificado na figura, representa as abordagens adoptadas nesta dissertação, que também serão justificadas mais à frente.

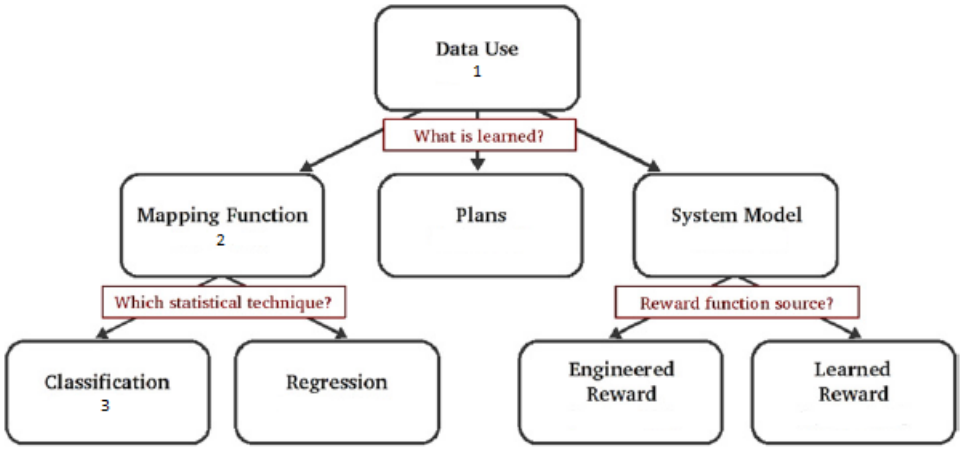


Figura 3.4: Categorização de abordagens para a aprendizagem de uma *policy*. Adaptado de Argall *et al.* [40].

A derivação de *policies* na abordagem Aprendizagem por Demonstração pode ser feita através de três métodos distintos: *mapping function*, *plans* e *system model*, tal como se encontra identificado na Figura 3.4. A primeira usa os dados directamente obtidos nas demonstrações para criar uma função que se aproxime do mapeamento feito entre os estados e as acções $f() : Z \; ! \; A$. Na segunda, a *policy* gerada corresponde a uma sequência de acções que guiam o agente desde o estado inicial até ao estado desejado. As acções são geralmente definidas através de pré e pós-condições $L(fpreC, postCgja)$. Ao contrário dos outros métodos da Aprendizagem por Demonstração, este planeamento da sequência de acções a tomar requer normalmente informações adicionais acerca das intenções do professor. Na última, as demonstrações são utilizadas na determinação de um modelo de transição de estados $T(s^0 js, a)$. Esta abordagem é tipicamente usada no âmbito da Aprendizagem por Reforço. Os dados provenientes das demonstrações e da exploração autónoma feita pelo sistema definem a função de transição. A este modelo pode-se acrescentar uma função de recompensa $R(s)$ (*reward*), cujo valor varia consoante a relevância do estado s em questão. O objectivo da Aprendizagem por Reforço é maximizar o valor cumulativo das recompensas, ao longo do tempo. Com estas informações combinadas é, então, derivada a *policy*.

A abordagem seleccionada neste trabalho para a geração de uma *policy* é a *mapping function*, uma vez que, como já foi referido anteriormente, o sistema pretende imitar as demonstrações do professor, sem ter que existir um planeamento prévio, com pré e pós-condições (*plans*), nem a criação de um modelo de transição dos estados do mundo (*system model*), que pode conduzir a várias imprecisões no sistema. Esta técnica considera-se determinística pois é-lhe dada toda a informação necessária para a elaboração da *policy*. O seu objectivo baseia-se no cálculo de uma *policy* que permita fazer uma generalização a partir do conjunto de demonstrações, de forma a adquirir soluções válidas também para estados que são semelhantes ou próximos dos estudados.

Na Figura 3.5 está representada esquematicamente a derivação da *policy* na abordagem *mapping function*. Como se pode observar, as demonstrações do professor correspondem a pares de exemplos de estado-acção, que ao passarem na técnica de aprendizagem, derivam uma *policy*. Esta consiste numa função que se aproxima do mapeamento estado-acção $\pi = f() : Z \ ! \ A$.

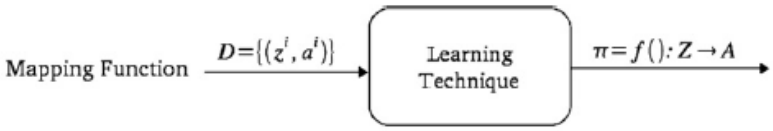


Figura 3.5: Derivação da *policy* através da *mapping function*. Fonte: Argall *et al.* [40].

A aproximação esperada pode ser influenciada por vários factores, entre os quais, se os estados e as acções correspondem a variáveis discretas ou contínuas, se a função de aproximação é para actuar antes do tempo de execução ou exactamente nesse momento, ou se se mantém o mesmo conjunto de demonstrações ao longo de todo o processo de aprendizagem. De um modo geral, as técnicas usadas na *mapping function* inserem-se em 2 categorias: *classification*, se a saída do algoritmo for discreta, e *regression*, se a saída for contínua. Ambas se encontram representadas na Figura 3.4. Os algoritmos de classificação categorizam os dados de entrada em classes discretas, tentando agrupar os valores semelhantes. No contexto deste trabalho, será utilizada esta abordagem uma vez que se pretende obter uma classificação dos dados de entrada, sendo que estes e a respectiva saída do algoritmo são dados discretos.

3.6 Algoritmos de classificação

Existem vários algoritmos de classificação que podem ser usados na abordagem referida anteriormente [41]. Entre eles constam os *Hidden Markov Models* (HMM) [42], *Conditional Random Fields* (CRF) [41], *Maximum Entropy Markov Models* (MEMM) [43], *k-Nearest Neighbors* (kNN) [44], entre outros. Estes algoritmos podem ser aplicados em diversas áreas científicas, tais como, a área da biologia computacional, do reconhecimento de gestos e no processamento da fala e de texto.

Segundo Lafferty *et al.* [41], os *Hidden Markov Models* são bastante utilizados em problemas onde é necessário segmentar e etiquetar sequências de dados. Esta abordagem é considerada um modelo generativo poderoso [45], que inclui estruturas de estados escondidos (*hidden states*). No entanto, sendo este um modelo generativo, existem algumas desvantagens no seu uso.

Os modelos generativos procuram aprender a distribuição de probabilidade conjunta, $p(x, y)$, sobre as observações x e as *labels* y, sendo factorizados da forma $p(x, y) = p(x)p(yjx)$ [46]. Para as suas previsões é calculada a probabilidade $p(yjx)$ e, posteriormente, é seleccionada a *label* y mais provável para uma determinada observação x, isto é, com a maior probabilidade [47]. Apesar destes modelos apresentarem vantagens, também revelam algumas limitações. Entre estas inclui-se a dificuldade em construir a distribuição de probabilidade sobre os dados, pois além das suas dimensões poderem ser elevadas, as suas *features* podem ter dependências complexas entre si. Embora modelar estas dependências dos dados de entrada possa gerar modelos intratáveis, ignorá-las pode levar a

um desempenho reduzido [46]. Adicionalmente, Lafferty *et al.* [41] referem que os modelos generativos precisam de enumerar todas as sequências de observações possíveis e não consideram prático a representação das interacções entre as *features*, do mesmo modo que, consideram difícil a introdução de dependências de longo alcance entre as observações.

Por outro lado, existem os modelos discriminativos, que também têm sido utilizados para resolver problemas de etiquetagem de sequências de dados. Neste caso, aprende-se a distribuição da probabilidade condicional $p(yjx)$ directamente, o que, na realidade, é tudo o que é necessário para a classificação [46, 47]. Assim, não é necessário gastar esforços em modelar as observações. Além disso, a probabilidade de transição pode depender não só da observação actual, como também das observações passadas ou futuras, quando disponíveis. Este facto contrasta com o realizado nos modelos generativos, que fazem suposições de independência bastante restritivas nas observações [41].

Portanto, a principal diferença entre os modelos generativo e discriminativo é que, para a distribuição condicional $p(yjx)$ do último, não é preciso gerar um modelo das observações $p(x)$, o que também não é necessário para a classificação [46]. Tal como já foi referido, é difícil modelar $p(x)$ pois podem existir dependências complexas entre as *features* das observações, que são difíceis de modelar.

O método *k-Nearest Neighbors* (kNN) é considerado das abordagens mais simples e antigas, utilizadas em *pattern classification* [44, 48]. Ainda assim, obtém resultados bastante competitivos comparativamente a outros algoritmos de classificação. É considerado um modelo discriminativo e o seu desempenho depende fundamentalmente da métrica da distância usada para identificar os seus vizinhos mais próximos. Por isso, esta métrica deve-se adaptar a cada problema individualmente [48]. Existem várias opções que podem ser usadas, tais como, a distância euclideana, a distância de Mahalanobis, a distância de Minkowski, a distância de Chebyshev, a distância de Manhattan (ou *city block*), a distância de Hamming, a distância de Jaccard e a distância de Spearman.

Os modelos *Maximum Entropy Markov Models* (MEMM) e *Conditional Random Fields* (CRF) também são discriminativos. A principal diferença entre ambos é que os MEMM utilizam distribuições exponenciais para modelar a probabilidade de transição para cada estado, enquanto que os CRF usam uma única distribuição exponencial para modelar toda a sequência de estados a partir do conjunto de observações [49]. Os *Conditional Random Fields* foram primeiramente introduzidos por Lafferty *et al.* [41], que referem que os CRF têm todas as vantagens dos MEMM e ainda resolvem o problema de *label bias* por ele apresentado. Além disso, se o CRF apresentasse uma dependência markoviana, seria uma versão discriminativa do algoritmo HMM.

Posto isto, neste trabalho, são utilizados os algoritmos kNN e CRF, pelas vantagens e simplicidade que apresentam, face aos outros algoritmos de classificação. Além destes, também é utilizado o algoritmo *Hidden Conditional Random Fields* (HCRF). Mais uma vez, trata-se de um modelo discriminativo e é considerado como uma extensão do CRF, que inclui uma camada de estados escondidos [45]. Este algoritmo pretende melhorar os resultados obtidos com o CRF, pois os seus dados de entrada são agrupados de forma a facilitar a sua etiquetagem. Nas próximas secções, são formalmente apresentados e explicados cada um destes 3 algoritmos.

3.6.1 *k-Nearest Neighbors* (kNN)

O método dos *k*-vizinhos mais próximos (em inglês, *k-Nearest Neighbors* (kNN)) classifica cada exemplo não etiquetado de acordo com as etiquetas dos seus *k*-vizinhos mais próximos, presentes no conjunto de treino [48].

Seja $D = f(x_i, y_i)g_{i=1}^{n}$ um conjunto de treino com n exemplos etiquetados, onde x_i corresponde a uma observação e y_i à sua respectiva *label*. Pretende-se fazer uma previsão das *labels* $\hat{y}_t$ que classificam um novo conjunto de dados $f(x_t)g_{t=1}^{m}$, denominado conjunto de teste, com tamanho m.

Para cada observação x_t do conjunto de teste, é calculada a distância $d(x_t, x_i)$ a todos os n exemplos do conjunto de treino, com a métrica de distância escolhida. De seguida, seleccionam-se os k exemplos mais próximos da observação x_t e contabiliza-se o número de vezes que apareceu cada uma das *labels*. A previsão da etiqueta $\hat{y}_t$, para a observação x_t, corresponde à *label* mais frequente nesse conjunto dos k-vizinhos mais próximos [48, 50].

As métricas de distância utilizadas neste trabalho foram a distância euclideana, a distância de Manhattan e a distância de Minkowski. No próximo capítulo será explicado o motivo que levou à sua selecção. As duas primeiras métricas indicadas correspondem a casos especiais da distância de Minkowski [51].

Dados dois pontos num espaço de dimensão n, $x = (x_1, ..., x_n)$ e $y = (y_1, ..., y_n)$, a distância de Minkowski é dada por

$$d_M(x, y) = \sqrt[p]{\sum_{i=1}^{n} jx_i \quad y_i j^p}, \tag{3.1}$$

sendo que, quando $p = 1$ tem-se à distância de Manhattan e quando $p = 2$ tem-se a distância euclideana. Portanto, a expressão matemática para o cálculo da distância euclideana [52] é dada por

$$d_E(x, y) = \sqrt{\sum_{i=1}^{n} (x_i \quad y_i)^2}. \tag{3.2}$$

A distância de Manhattan também é conhecida como distância *city block* ou *taxicab metric*. Considerando, mais uma vez, os pontos x e y, num espaço de dimensão n, a distância *city block* [53] calcula-se através da expressão

$$d_{cb}(x, y) = \sum_{i=1}^{n} jx_i \quad y_i j. \tag{3.3}$$

3.6.2 *Conditional Random Fields* (CRF)

O objectivo do algoritmo CRF é prever uma classificação para um determinado conjunto de dados de entrada. Neste método é calculado directamente a probabilidade condicional de uma *label* (ou sequência de *labels*) classificar uma determinada observação (ou sequência de observações). Não existe, por isso, a necessidade de calcular a probabilidade marginal das observações [41].

Seja x uma sequência de observações e y a sequência de *labels* que a classifica. Ambas as variáveis têm o mesmo tamanho n e definem-se por $x = x_1, x_2, ..., x_n$ e $y = y_1, y_2, ..., y_n$, respectiva-

mente [54]. Cada *label* y_i classifica uma observação x_i e corresponde a um elemento do conjunto de *labels* possíveis γ, isto é, y_i *2* γ. Neste trabalho, γ corresponde ao conjunto de *labels* discriminado na Tabela 3.2, isto é, as *labels* de 1 a 7.

Neste algoritmo, introduz-se o conceito de *feature function*, cujo objectivo é expressar algum tipo de relação ou característica nas sequências de entrada. Geralmente, as *features* dependem dos dados de entrada em torno de uma posição específica. No entanto, também podem depender de propriedades ou características globais dos mesmos [54]. Segundo Morency *et al.* [55], as *feature functions* podem ser de dois tipos diferentes: *transition feature function* que tem em consideração o estado anterior e o estado actual e *state feature function* que se foca somente no estado actual. Todas as *feature functions* assumem valores reais. Contudo, tomam normalmente os valores 0 ou 1, sendo 1 quando apresentam a *feature* específica e 0 caso contrário [46].

Segundo Lafferty *et al.* [41], a probabilidade de uma sequência de *labels* y etiquetar uma sequência de observações x conhecida tem a forma de

$$p_\theta(y \mid x) \propto exp\left(\sum_j \lambda_j t_j(y_{i-1}, y_i, x, i) + \sum_k \mu_k s_k(y_i, x, i)\right), \tag{3.4}$$

onde $t_j(y_{i-1}, y_i, x, i)$ corresponde à *transition feature function* e $s_k(y_i, x, i)$ representa a *state feature function*. A variável i define uma determinada posição nas sequências indicadas e as variáveis λ_j e μ_k dizem respeito aos parâmetros a serem estimados pelo algoritmo a partir do conjunto de treino, constituindo o vector $\theta = (\lambda_1, \lambda_2, ...; \mu_1, \mu_2, ...)$.

Para facilitar a notação, considera-se que $s(y_i, x, i) = s(y_{i-1}, y_i, x, i)$, permitindo, deste modo, gerar um *global feature vector* [54] dado por

$$F_w(y, x) = \sum_{i=1}^{n} f_w(y_{i-1}, y_i, x, i), \tag{3.5}$$

onde cada $f_w(y_{i-1}, y_i, x, i)$ pode corresponder a uma *state feature function* $s(y_{i-1}, y_i, x, i)$ ou a uma *transition feature function* $t(y_{i-1}, y_i, x, i)$ [46]. O somatório em i pretende englobar todas as observações.

Portanto, pelo algoritmo CRF [56], a probabilidade de uma sequência de *labels* y etiquetar uma conhecida sequência de observações x pode ser escrita como

$$p(y \mid x, \theta) = \frac{1}{Z(x)} exp\left\{\sum_w \theta_w F_w(y, x)\right\}, \tag{3.6}$$

onde $Z(x)$ corresponde à função de normalização dada por

$$Z(x) = \sum_y exp\left\{\sum_w \theta_w F_w(y, x)\right\}. \tag{3.7}$$

É de notar que o somatório em w pretende abranger todas as *feature functions*. A função de normalização $Z(x)$ é também conhecida como *partitioning function*.

A sequência de *labels* mais provável $\hat{y}$ de classificar correctamente uma determinada sequência de

dados x resulta da expressão

$$\hat{y} \; 2 \; arg \; max_y \; p(y \; j \; x, \theta) = arg \; max_y \sum_w \theta_w F_w(y, x), \tag{3.8}$$

sendo que se pretende maximizar o número de *labels* correctas na sequência de observações x. A função $Z(x)$ pode ser ignorada pois executa apenas a normalização dos valores das probabilidades e não depende de y [54]. A função exponencial também é ignorada uma vez que não altera o valor do $argmax$. Posto isto, falta apenas determinar o parâmetro θ que se obtém a partir do treino do algoritmo com um conjunto de dados etiquetados. Este parâmetro corresponde a um vector de pesos que avalia a relevância de cada *feature*, atribuindo-lhes um peso.

Seja $D = f(x_i, y_i)g_{i=1}^n$ um conjunto de dados de treino com n observações etiquetadas. No treino do algoritmo CRF, é normalmente utilizado o método da máxima verosimilhança, que selecciona os valores para o parâmetro θ, que maximizam o logaritmo da verosimilhança, ou *log-likelihood*, do conjunto de treino [54]. Tem-se, portanto,

$$L(\theta) = log \; p(y \; j \; x, \theta) = \sum_w \theta_w F_w(y, x) \quad logZ(x) \tag{3.9}$$

e

$$\hat{\theta} \; 2 \; arg \; max_\theta \; L(\theta) = arg \; max_\theta \; log \; p(y \; j \; x, \theta). \tag{3.10}$$

Uma das formas de calcular $\hat{\theta}$ é através da expressão

$$\frac{\partial L(\theta)}{\partial \theta} = 0. \tag{3.11}$$

Contudo, nem sempre é possível calcular analiticamente os valores dos parâmetros que maximizam o *log-likelihood* [56]. Deste modo, é necessário recorrer a métodos iterativos, tais como, o *gradient descent*, o *Newton-Raphson* ou os métodos *Quasi-Newton*. Neste trabalho foi utilizado o algoritmo *Broyden-Fletcher-Goldfarb-Shanno* (BFGS) que pertence à família dos métodos *Quasi-Newton*. A determinação do parâmetro $\hat{\theta}$ é, então, feita através da inicialização em vários pontos aleatórios [57].

3.6.3 *Hidden Conditional Random Fields* (HCRF)

No caso do algoritmo *Conditional Random Fields*, as sequências de *labels* são consideradas como totalmente observáveis para os dados de treino. Contudo, na abordagem *Hidden Conditional Random Fields*, além destas *labels* observáveis, é aprendida uma camada adicional intermediária. Estas novas variáveis intermediárias modelam a parte latente dos dados de entrada e são chamadas de variáveis (ou *labels*) escondidas. O modelo calcula a probabilidade conjunta das *labels* e das *labels* escondidas condicionadas às observações, tendo em consideração a existência de dependências entre as variáveis escondidas [57].

Seja $D = f(x_i, y_i)g_{i=1}^n$ um conjunto de dados de treino com n observações etiquetadas, onde cada exemplo, x_i, corresponde a um vector de m observações locais, sendo $x_i = x_{i,1}, x_{i,2}, ..., x_{i,m}$.

Cada observação local, x_{ij}, é representada pelo *feature vector* $\phi(x_{ij})$ *2* $<^d$, sendo d a dimensão da representação [57]. Cada *label* y_i classifica um exemplo x_i e corresponde a um elemento do conjunto de *labels* possíveis γ, ou seja, y_i *2* γ. Mais especificamente, o conjunto γ utilizado neste trabalho corresponde às *labels* descritas na Tabela 3.2.

Para cada observação x_i, assume-se também um vector de variáveis latentes $h_i = h_{i,1}, h_{i,2}, ..., h_{i,r}$, que não são observadas no conjunto de treino. No máximo, este vector h_i tem tamanho m, ou seja, $r = m$, que corresponde ao número de observações locais. A variável *nbHiddenStates*, que será utilizada posteriormente neste trabalho (na secção 4.4), é r e refere-se ao número de estados escondidos. Cada h_{ij} corresponde a um elemento de H, sendo H um conjunto finito de possíveis *labels* escondidas no modelo. Segundo Quattoni *et al.* [57], de forma intuitiva, cada h_{ij} corresponde à *label* de x_{ij} conjugada com algum elemento de H, que pode ser, por exemplo, uma parte da estrutura de x_{ij}.

Neste algoritmo, as *feature functions* podem ser de 3 tipos diferentes [55, 58]. Além das duas descritas anteriormente para o algoritmo CRF, com a introdução de *hidden states*, surge outra *feature function*. Portanto, a *state feature function* depende apenas da observação actual e da sua posição no vector de observações, sendo que cada *feature* é criada para cada par observação-*hidden state*, ou seja, relaciona x_{ij} com h_{ij}. A *transition feature function* verifica se existe ligação entre dois *hidden states* adjacentes. A terceira *feature function*, que não existia no CRF, relaciona um *hidden state* (ou *label* escondida) com a sequência das *labels* [55].

A partir destas definições do conjunto total de observações x, da sua sequência de *labels* y e das *labels* escondidas h, a probabilidade condicional das *labels* dadas as observações é escrita da forma

$$p(y \mid x, \theta) = \sum_h p(y, h \mid x, \theta) = \frac{\sum_h exp\{\Psi(y, h, x; \theta)\}}{\sum_{y,h} exp\{\Psi(y, h, x; \theta)\}}, \tag{3.12}$$

onde θ corresponde ao *parameter vector*, semelhante ao do algoritmo CRF, e $\Psi(y, h, x; \theta)$ *2* $\mathbb{R}$ diz respeito à *potential function*. O denominador da equação 3.12 tem o nome de *partition function* e o seu objectivo é assegurar a normalização das probabilidades no modelo.

Relativamente à *potential function*, esta define-se por

$$\Psi(y, h, x; \theta) = \sum_j \sum_l f_l^1(j, y, h_j, x)\theta_l^1 + \sum_{j,k} \sum_l f_l^2(j, k, y, h_j, h_k, x)\theta_l^2, \tag{3.13}$$

onde f_l^1 e f_l^2 correspondem às *feature functions* que definem as *features* do modelo e θ_l^1 e θ_l^2 constituem o *parameter vector* θ. Os somatórios em l pretendem englobar todas as *feature functions* existentes. As *features* de f^1 dependem de valores únicos das variáveis escondidas, enquanto que as *features* de f^2 dependem de pares de valores de *hidden labels*, existindo, por isso, os dois índices j e k [57]. Posto isto, a expressão f_l^1 pode-se decompor em duas funções, uma delas expressa a compatibilidade entre as *labels* y e os estados escondidos h_j (referida anteriormente como a terceira *feature function*) e a outra entre as observações x_j e os estados escondidos h_j (*state feature function*) [58]. Desta forma, é perceptível que a função f^2 diz respeito à *transition feature function*, já explicada anteriormente.

A sequência de *labels* $\hat{y}$ mais provável de classificar correctamente uma nova sequência de dados x (conjunto de teste), resulta da expressão

$$\hat{y} \; 2 \; arg \; max_{\; y} \; p(y \; j \; x, \theta \;), \tag{3.14}$$

onde θ corresponde ao *parameter vector* obtido no treino do modelo. Segundo Wang *et al.* [45], a estimação de θ pode ser feita através da maximização da função

$$L(\theta) \; 2 \sum_{i=1}^{n} log \; p(y_i \; j \; x_i, \theta) \quad \frac{1}{2\sigma^2} k\theta k^2, \tag{3.15}$$

com o conjunto de treino D, já definido anteriormente, sendo que

$$\theta \;\; 2 \; arg \; max_{\theta} \; L(\theta). \tag{3.16}$$

O primeiro termo da equação 3.15 corresponde ao *log-likelihood* dos dados e o segundo termo corresponde ao logaritmo do *Gaussian prior* com variância σ^2, ou seja, $p(\theta) \quad exp\left(\frac{1}{2\sigma^2} k\theta k^2\right)$ [57]. Para calcular os valores óptimos dos parâmetros θ podem ser utilizados vários algoritmos de optimização não-lineares. De forma análoga ao realizado anteriormente, foi usado o algoritmo *Broyden-Fletcher-Goldfarb-Shanno* que pertence à família dos métodos *Quasi-Newton*. A determinação do parâmetro θ é também feita através da inicialização em vários pontos aleatórios e procurando o melhor máximo local [57].

3.7 Implementação

Os dados do jogo Rabelos, referidos na secção 3.4, juntamente com os dados da posição do jogador, indicados na secção 3.2, são reunidos num ficheiro .csv, obtido directamente do sistema PEPE. Este ficheiro está organizado por tempo e contém 60 leituras por segundo, ou seja, cada leitura abrange um intervalo de, aproximadamente, 17 milissegundos do jogo.

Relembrando que o jogo Rabelos era o que continha mais amostras dos *feedbacks* destacados (rever secção 3.4), complementa-se agora que são estes 7 conjuntos de *feedback* em destaque, os que serão "replicados" pelo sistema, pois é conveniente que existam dados em quantidade suficiente para aplicar os algoritmos. Portanto, os *feedbacks* seleccionados para reproduzir, neste trabalho, encontram-se sintetizados, e novamente etiquetados, com números de 1 a 7, na Tabela 3.2.

Para alinhar as mensagens de *feedback* transcritas com os jogos, foi necessário anotar o instante de tempo em que estas mensagens foram proferidas, de preferência na mesma ordem de grandeza em que são efectuadas as leituras dos dados, isto é, em milissegundos. Para isso, foi utilizado o programa *Wavesurfer*, que permite a análise do tempo na unidade de medida pretendida. Registou-se um intervalo de 5 milissegundos em que a mensagem começou a ser dita. Por exemplo, a Figura 3.6 mostra a forma de onda da expressão "vamos lá". Como é possível observar, esta expressão começou a ser dita no intervalo de tempo assinalado a cor amarela, entre 1:16.20 e 1:16.25.

Tabela 3.2: Identificação dos *feedbacks* utilizados neste trabalho, com a respectiva contagem.

Label	***Feedbacks***	**Contagem**
1	Bora Bora lá Vamos embora	134
2	Continue Continua a remar Continue em frente	171
3	Reme/Rema Remar Rodar Reme bem Temos que remar Tem que remar	51
4	Vamos lá Vá lá Vamos remar Vamos a isso	211
5	Isso mesmo É isso mesmo É isso Assim é que é	51
6	Boa	152
7	Isso	116

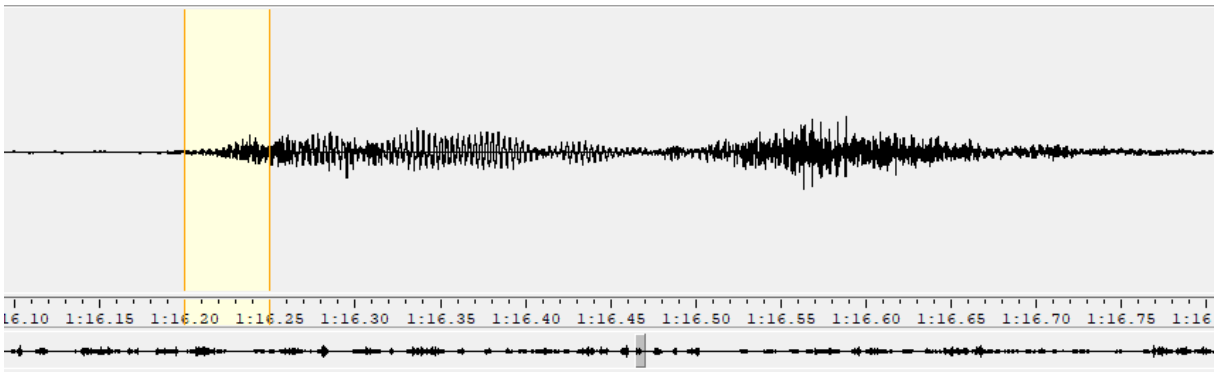


Figura 3.6: Forma de onda da expressão “vamos lá”, obtida através do *Wavesurfer*.

Foram anotados todos os instantes de tempo para cada um dos *feedbacks*, pois era fulcral a existência de um alinhamento entre os dados dos ficheiros .csv e as mensagens de *feedback* das gravações de vídeo.

Além disso, nem todos os parâmetros da posição do jogador e do próprio jogo foram utilizados para realizar esta replicação. Uma vez que o jogo Rabelos tem o principal foco nos membros superiores, devido às acções de remar e de recolha de barris, e ainda como grande parte dos jogos foram realizados com o jogador sentado, foram escolhidos os parâmetros com incidência nos braços, ignorando os relativos aos membros inferiores. Portanto, as articulações do jogador escolhidas foram as mãos, os pulsos e os cotovelos de ambos os lados, direito e esquerdo. Para cada uma das articulações foram consi-

deradas as suas coordenadas XYZ em metros (*PositionX*, *PositionY*, *PositionZ*) e a sua rotação em relação ao referencial ilustrado na Figura 3.1 (*OrientationX*, *OrientationY*, *OrientationZ*, *OrientationW*). Assumiu-se que as outras articulações não causariam tanto impacto, pois não estariam em constante movimento durante o jogo e, por isso, foram desprezadas.

No que diz respeito aos parâmetros próprios do jogo, apenas foram considerados o número de remadas dadas até ao momento (*Rows*) e o número de barris recolhidos até ao instante em causa (*Barrels*). De forma intuitiva, consegue-se associar o parâmetro *Barrels* aos indicadores *PickingLeft* e *PickingRight*, pois sempre que um destes muda para o valor 1, incrementa o número de barris, e o número de remadas está directamente relacionado com o indicador *MovingForward*. Os outros parâmetros não mostram tanta relevância na caracterização do remar, pelo que também não foram considerados.

Agrupando todas estas informações, foi possível gerar ficheiros de entrada específicos para o pretendido. Este projecto foi desenvolvido em MATLAB e na próxima secção encontra-se descrito todos os detalhes da sua implementação.

3.7.1 Código MATLAB

Para implementar os algoritmos CRF e HCRF recorreu-se à biblioteca HCRF, disponível no site SourceForge [59]. Além dos *scripts* fornecidos por esta biblioteca, para o treino e o teste destes dois algoritmos, foram também produzidos outros *scripts* e funções, incluindo um ficheiro para a aplicação do algoritmo kNN. A organização do código encontra-se representada na Figura 3.7.

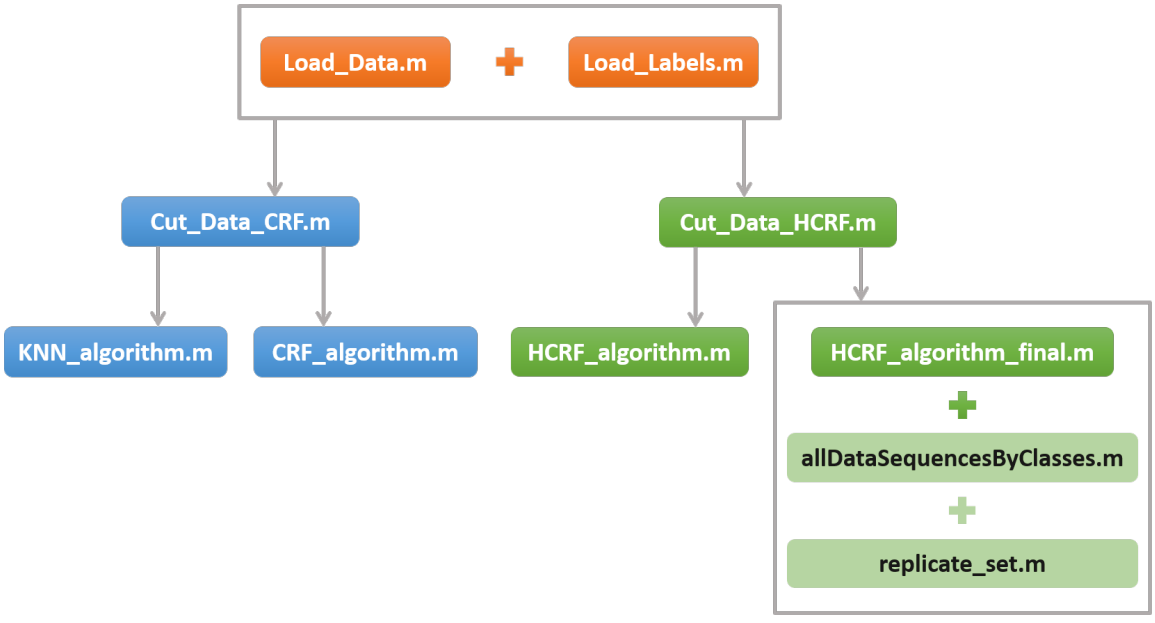


Figura 3.7: Representação esquemática dos scripts desenvolvidos.

Numa primeira fase, é necessário unir os dados dos jogos e das *labels* de todos os vídeos, tendo o mesmo sido realizado nos *scripts* Load_Data.m e Load_Labels.m, respectivamente. Os algoritmos requerem uma organização dos dados de entrada específica e, por isso, foram criados 2 ficheiros

distintos, Cut_Data_CRF.m e Cut_Data_HCRF.m, que permitem preparar os dados da forma correcta. O algoritmo kNN também utiliza o ficheiro Cut_Data_CRF.m, pois a organização dos seus dados é semelhante à do algoritmo CRF. Por conseguinte, foi possível desenvolver os *scripts* onde se aplicam cada um dos algoritmos pretendidos.

Mais especificamente para o algoritmo HCRF, foram gerados 2 ficheiros cuja diferença incide no conjunto de dados utilizados no treino do classificador. No ficheiro HCRF_algorithm.m, os dados usados para treinar são exactamente aqueles que saem do ficheiro Cut_Data_HCRF.m. Por outro lado, no HCRF_algorithm_final.m, os dados utilizados no treino são replicados de modo a existir um equilíbrio no número de amostras de cada *label*. Por exemplo, se num conjunto de dados etiquetados com apenas 2 *labels*, houver 50 etiquetados com a *label* A e 80 com a *label* B, então os dados da *label* A são replicados até perfazerem 80. Resulta um total de 160 dados (80 da *label* A + 80 da *label* B), em vez dos anteriores 130 utilizados em HCRF_algorithm.m (50 da *label* A + 80 da *label* B). Para esta replicação foram desenvolvidas 2 funções que permitiram auxiliar o processo. Da função allDataSequencesByClasses.m obtém-se um *cell array* com os dados organizados por *label*. Quer isto dizer que desta função resulta um *cell array* de tamanho 1x7, sendo que 7 corresponde ao número de *labels*, e cada uma destas células possui 1xN *cell arrays*, sendo N o número de amostras etiquetadas com essa *label* em questão. A função replicate_set.m executa a dita replicação dos dados, utilizando o *cell array* obtido na função allDataSequencesByClasses.m.

É de notar ainda que, em todos os ficheiros KNN_algorithm.m, CRF_algorithm.m, HCRF_algorithm.m e HCRF_algorithm_final.m, foi utilizado o método de validação cruzada (em inglês, *cross-validation*) designado *k-fold* [60]. Neste método, o conjunto total de dados é dividido em k subconjuntos, mutuamente exclusivos, sendo que um desses subconjuntos é utilizado para o teste e os restantes, $k \quad 1$, são utilizados no treino. O subconjunto de teste vai variando de modo a percorrer todo o conjunto de dados, isto é, os k subconjuntos. Tal é conseguido através de um ciclo *for* com k iterações. No fim de cada iteração calcula-se a *accuracy* dos dados obtidos no teste, sendo que no fim de todas as iterações calcula-se a média das *accuracies* e o seu desvio padrão. Este método permite avaliar a precisão do classificador e comparar o desempenho dos algoritmos usados.

À medida que o código foi desenvolvido, verificou-se que um dos ficheiros de dados se encontrava corrompido, pelo que não pôde ser utilizado nos testes. Desta forma, ficaram apenas, para estudo, 39 ficheiros do jogo Rabelos. Foi feita uma nova contagem dos *feedbacks* proferidos nestes 39 jogos, encontrando-se na Tabela 3.3 os resultados obtidos. É de notar que esta tabela é uma reformulação da Tabela 3.2, indicada anteriormente.

De seguida, são explicados com mais detalhe os 8 *scripts* desenvolvidos neste trabalho.

Load_Data.m

Deste *script* resulta uma variável *cell array* de tamanho 1x39, sendo 39 o número de execuções do jogo Rabelos. Cada uma destas células contém os dados de todos os instantes de tempo para os parâmetros seleccionados (rever secção 3.7). Por exemplo, uma das células tem o tamanho de 10000x45. Neste caso, 10000 corresponde ao número de leituras feitas e 45 (44+1) refere-se aos 44 parâmetros esco-

Tabela 3.3: Reformulação da contagem dos *feedbacks* utilizados.

Label	**Contagem**
1	128
2	161
3	50
4	210
5	45
6	148
7	109

lhidos, mais uma coluna indicando o instante de tempo dessa leitura, que será, posteriormente, usado para fazer o alinhamento com as *labels*. O tamanho 10000 pode variar de célula para célula, pois nem todos os jogos tiveram a mesma duração. Por outro lado, o tamanho 45 mantém-se constante, porque analisaram-se sempre os mesmos parâmetros.

Load_Labels.m

Neste *script* gera-se uma variável *cell array* de tamanho 1x39, sendo que 39 é o número de vídeos do jogo Rabelos, como indicado anteriormente. Cada uma das células contém as *labels* referidas nesse jogo e o respectivo instante de tempo. Esse instante corresponde ao fim do intervalo de 5 milissegundos anotado na secção 3.7, de modo a assegurar que o *feedback* já tinha começado a ser dito.

Cut_Data_CRF.m

Este *script* utiliza as variáveis que resultam de ambos os ficheiros Load_Data.m e Load_Labels.m. Aqui define-se o intervalo de tempo a considerar antes e depois do instante de tempo de cada *label*. Por exemplo, pode-se analisar os 300 milissegundos anteriores ao instante da *label* e os 100 milissegundos posteriores. Este intervalo é explicado mais pormenorizadamente na secção 4.3.1.

É também realizado um rearranjo nos dados para o formato correcto de entrada no algoritmo CRF. Deste *script* resultam 2 variáveis *cell array*, uma com os dados e outra com as *labels*.

Cut_Data_HCRF.m

Semelhante ao *script* anterior, são utilizados os resultados dos ficheiros Load_Data.m e Load_Labels.m e é definido o intervalo de tempo que engloba o instante de cada *label*.

É feito um rearranjo nos dados para se encontrarem no formato correcto para o algoritmo HCRF, que difere do formato de entrada do algoritmo CRF. Deste *script* também resultam 2 variáveis *cell array*, uma com os dados e outra com as *labels*.

KNN_algorithm.m

Neste *script* é utilizado o algoritmo kNN para o treino do classificador. Os dados de entrada são obtidos a partir do ficheiro Cut_Data_CRF.m, com uma ligeira alteração na estrutura das variáveis, pois são necessárias as suas transpostas. No treino do classificador é utilizada a função *fitcknn* do MATLAB. Esta permite optimizar os hiperparâmetros automaticamente, ou seja, para utilizar o algoritmo kNN é necessário fornecer o número de vizinhos k e a métrica da distância, que devem ser óptimos de modo a gerarem o melhor classificador possível, e esta função consegue encontrar esses parâmetros óptimos automaticamente.

Foi utilizado o método *k-fold*, explicado anteriormente, com $k = 3$. Em cada iteração, é feito o cálculo da *accuracy* do conjunto de teste e de cada *label*, sendo que no final determinam-se as médias e os desvios padrão destas *accuracies*.

CRF_algorithm.m

Neste *script* recorre-se ao algoritmo CRF para o treino do classificador, a partir dos dados obtidos directamente do *script* Cut_Data_CRF.m. É escolhido o valor do parâmetro *windowSize*, que define a quantidade de observações, passadas e futuras, que serão consideradas na previsão de uma observação específica, conseguindo desta forma incorporar dependências de longo alcance entre as observações [45]. Também foi utilizado o método *k-fold*, explicado anteriormente, com $k = 3$. Em cada iteração, é calculada a *accuracy* do conjunto de teste e a *accuracy* de cada *label*, sendo que no final determinam-se as suas médias e os respectivos desvios padrão.

HCRF_algorithm.m

Neste *script* é utilizado o algoritmo HCRF no treino do classificador. Os dados utilizados são directamente as variáveis resultantes do *script* Cut_Data_HCRF.m. Também foi usado o método *k-fold* com $k = 3$ e definiram-se os valores de *windowSize* e de *nbHiddenStates*. Em cada iteração, é feito o cálculo da *accuracy* do conjunto de teste e da *accuracy* de cada *label*. No final, determina-se a média e o desvio padrão de ambas as *accuracies* calculadas.

HCRF_algorithm_final.m

Neste *script* também é utilizado o algoritmo HCRF para o treino do classificador. Os dados utilizados são as variáveis resultantes do *script* Cut_Data_HCRF.m. Contudo, após a definição dos conjuntos de teste e de treino, este último sofre uma alteração no seu conteúdo, sendo-lhe aplicado ambas as funções allDataSequencesByClasses.m e replicate_set.m. Esta reformulação pretende equilibrar o número de amostras por *label*, tal como fora referido anteriormente.

De modo análogo aos *scripts* anteriores, foi usado o método *k-fold*, com $k = 3$, e definiram-se os parâmetros *windowSize* e *nbHiddenStates*. Em cada iteração, é feito o cálculo da *accuracy* do conjunto de teste, assim como, da *accuracy* de cada *label*. No final, determina-se a média e o desvio padrão destas *accuracies*.

Capítulo 4

Resultados e Discussão

Neste capítulo estão descritos todos os testes realizados e os respectivos resultados obtidos, com os 4 ficheiros KNN_algorithm.m, CRF_algorithm.m, HCRF_algorithm.m e HCRF_algorithm_final.m. Estes ficheiros aplicam os algoritmos de classificação kNN, CRF e HCRF, descritos anteriormente na secção 3.6. A escolha dos parâmetros testados é explicada e procura-se uma justificação para os resultados obtidos. Por fim, realiza-se uma comparação entre os vários resultados conseguidos, bem como, dos que se estavam à espera de obter.

4.1 Pré-processamento dos dados

Para a concretização dos testes, foi necessário gerar primeiramente os ficheiros de dados a estudar. Estes foram conseguidos a partir dos ficheiros de código Cut_Data_CRF.m e Cut_Data_HCRF.m, explicados anteriormente na secção 3.7.1.

No caso dos algoritmos kNN e CRF foram desenvolvidos ficheiros de dados que se podem inserir em 3 partes distintas. Na primeira parte, foram seleccionados intervalos de tempo com o mesmo valor antes e depois do instante de tempo que define o início do *feedback*. A gama de valores testados vai desde os 200 milissegundos até aos 2000 milissegundos, de modo a ser mais abrangente. Não se considerou necessário testar valores inferiores a 200 milissegundos, devido ao tempo de reacção dos fisioterapeutas, ou seja, desde que o idoso executa um determinado movimento até o fisioterapeuta dar a sua resposta, existe um certo intervalo que deve ser considerado. O tempo de reacção de uma pessoa pode variar entre os 250 e os 350 milissegundos, aproximadamente [61]. Contudo, este valor varia consoante as condições do ambiente envolvente e com a própria pessoa em si. Por outro lado, assumiu-se que o fisioterapeuta não ia demorar mais de 2 segundos a reagir ao comportamento do idoso, pelo que também não se justifica testar valores superiores a 2 segundos. Posto isto, os valores analisados, nesta parte, foram 200, 250, 300, 350, 500, 1000, 1500 e 2000 milissegundos, antes e depois do instante de tempo que define o início de cada *feedback*.

Na segunda parte, alterou-se o intervalo de tempo a testar. Uma vez que, quando os *feedbacks* são proferidos, referem-se a algo que já aconteceu, ou seja, são uma resposta do que ocorreu no

jogo ou com o jogador, analisa-se agora apenas os intervalos de tempo que antecedem o *feedback*. Manteve-se, portanto, o mesmo conjunto de valores de 200, 250, 300, 350, 500, 1000, 1500 e 2000 milissegundos anteriores ao instante de tempo que marca o início do *feedback* e 0 milissegundos no intervalo posterior a esse instante.

Na terceira parte, teve-se em consideração o erro que pôde existir quando foi feito o alinhamento entre os ficheiros de vídeo dos jogos e as mensagens provenientes das transcrições. Uma vez que, cada ficheiro de vídeo não contém apenas o desenrolar do jogo, mas também inclui os momentos de preparação que o antecedem, foi necessário encontrar o instante de tempo exacto que definisse o início do jogo. Somente a partir da definição deste instante inicial, seria possível anotar os tempos dos vários *feedbacks* dados no decorrer do mesmo. Todavia, os programas utilizados para visualizar os vídeos[1] mostravam apenas *frames* com uma diferença de 200 a 300 milissegundos, aproximadamente. Por essa razão, assumiu-se um erro de 300 milissegundos, que pode ter existido quando se estabeleceu o instante de início de cada jogo. O *delay* deste instante inicial provoca também um *delay*, da mesma dimensão, em cada um dos instantes de tempo definidos para os *feedbacks*. Posto isto, os intervalos de tempo analisados, nesta parte, observam apenas os 300 milissegundos posteriores a esse instante. Relativamente aos valores que o antecedem, foram considerados os intervalos 300, 500, 750, 1000 e 1500 milissegundos, nunca inferiores ao erro considerado.

Para o algoritmo HCRF, foi utilizado o ficheiro de código Cut_Data_HCRF.m, para criar apenas dois ficheiros de dados, com 300 e 500 milissegundos anteriores ao instante que marca o início de cada *feedback*, e com 300 milissegundos posteriores a esse instante. Considera-se relevante ter em atenção o erro, dos 300 milissegundos, que pôde existir no alinhamento dos instantes iniciais de cada jogo.

4.2 Algoritmo kNN

4.2.1 Testes com o ficheiro KNN_algorithm.m

Para a aplicação do algoritmo kNN nos dados dos jogos e dos seus *feedbacks*, foram utilizados os ficheiros de dados descritos anteriormente na secção 4.1. Estes ficheiros encontram-se distribuídos em 3 partes distintas, onde se têm em atenção diferentes considerações.

4.2.2 Resultados e discussão do obtido em KNN_algorithm.m

Os resultados obtidos com o algoritmo kNN encontram-se sintetizados na Tabela 4.1, onde é apresentada a média das *accuracies* e o respectivo desvio padrão, com um arredondamento a 4 casas decimais. Os resultados que irão aparecer nas secções seguintes também terão esta aproximação. É importante relembrar que é feita a média devido ao método *k-fold* aplicado.

A função *fitcknn* do MATLAB, que determina a métrica da distância mais adequada, mostrou que, em 92% dos testes efectuados, essa distância era a *city block*. Os restantes 8% dividem-se em 5%

[1] GOM Player e VLC media player.

Tabela 4.1: Resultados da *accuracy*, com o código KNN_algorithm.m, para os vários ficheiros indicados.

	Intervalos de tempo		**Média da *accuracy***	**Desvio padrão da *accuracy***
	Antes (ms)	**Depois (ms)**		
Parte 1	200	200	0.1890	0.0133
	250	250	0.1873	0.0078
	300	300	0.1866	0.0085
	350	350	0.1803	0.0047
	500	500	0.1813	0.0036
	1000	1000	0.1879	0.0031
	1500	1500	0.1862	0.0059
	2000	2000	0.1880	0.0030
Parte 2	200	0	0.1874	0.0104
	250	0	0.1726	0.0080
	300	0	0.1816	0.0070
	350	0	0.1788	0.0111
	500	0	0.1793	0.0119
	1000	0	0.1885	0.0120
	1500	0	0.1822	0.0087
	2000	0	0.1781	0.0100
Parte 3	500	300	0.1835	0.0069
	750	300	0.1848	0.0062
	1000	300	0.1872	0.0092
	1500	300	0.1855	0.0041

para a distância euclideana e 3% para a distância de Minkowski. No que diz respeito ao número de k vizinhos a considerar, obteve-se sempre o mesmo valor de $k = 1$.

Para facilitar a análise destes resultados, foi gerado o gráfico da Figura 4.1, que tem no eixo das abcissas os intervalos de tempo utilizados e no eixo das ordenadas as médias das *accuracies*. Para simplificar a descrição dos intervalos de tempo, define-se o nome de cada ficheiro como o conjunto de dois valores [X, Y], sendo X o intervalo anterior ao início do *feedback* e Y o intervalo posterior. O desvio padrão encontra-se representado pelas barras verticais, associadas a cada ponto do gráfico, e o seu comprimento encontra-se na mesma escala das *accuracies*. As duas barras verticais amarelas, a tracejado, separam as 3 partes estipuladas. Na terceira parte não foi representado o resultado do [300,300]ms uma vez que já estava repetido na Parte 1.

Tal como é possível observar os valores das *accuracies* são relativamente próximos, variando num intervalo entre 0.1726 e 0.1890. Os desvios padrão são considerados baixos, comparativamente às médias obtidas, atingindo no máximo o valor de 0.0133. Tanto na Parte 1 como na Parte 3, obtiveram-se sempre valores de *accuracy* superiores a 0.18, o que mostra a possível relevância do intervalo de tempo posterior ao início do *feedback*. As *accuracies* mais elevadas, em cada uma das partes, foram obtidas quando o intervalo anterior ao instante do *feedback* toma os valores de 200 ou de 1000 milissegundos.

Além disso, é também necessário analisar as *accuracies* obtidas para cada uma das *labels* nos vários ficheiros de dados. Neste sentido, apresenta-se a Tabela B.1 no Anexo B que indica não só os valores dessas *accuracies* como o seu desvio padrão. De um modo geral, verifica-se que cada uma das *labels* tem *accuracies* bastante idênticas, independentemente dos intervalos de tempo considerados em cada ficheiro. As *accuracies* das *labels* 3 e 5 destacam-se pelos seus valores baixos comparativamente

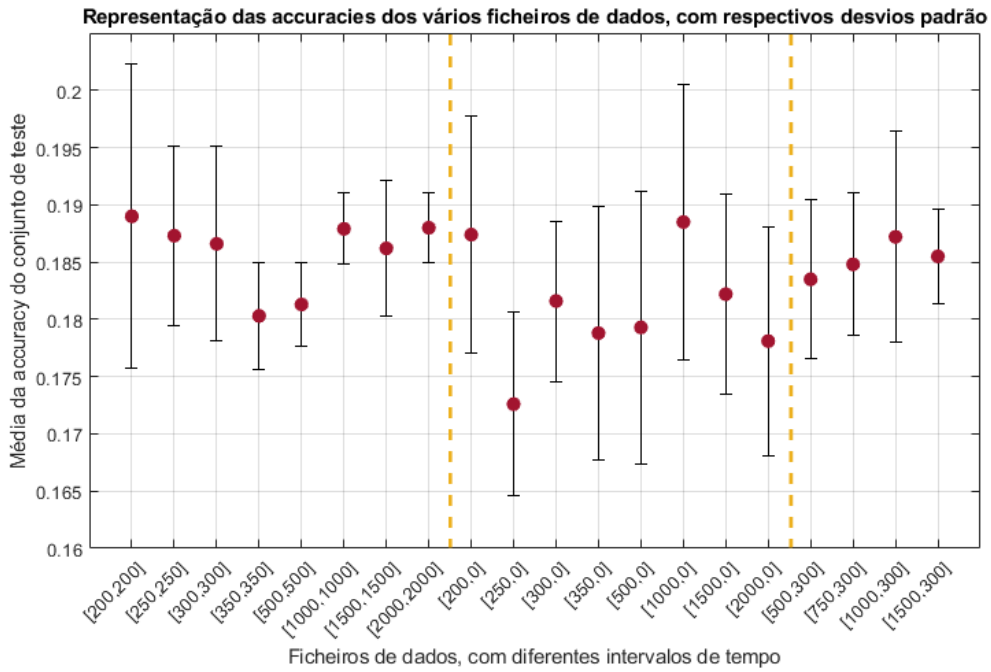


Figura 4.1: Representação gráfica de todos os valores das *accuracies* obtidos, com o algoritmo kNN.

às outras *labels*, tendo uma média de 0.056 e 0.034, respectivamente. Por outro lado, a *label* 4 é aquela que atinge valores mais elevados, variando em torno dos 0.307. As restantes *labels* apresentam uma gama de valores sensivelmente idêntica.

4.3 Algoritmo CRF

4.3.1 Testes com o ficheiro CRF_algorithm.m

Na aplicação do algoritmo CRF, foi utilizada a mesma gama de valores para os intervalos de tempo considerados no algoritmo kNN (rever secção 4.1). Adicionalmente, foram testados 3 valores diferentes para o parâmetro *windowSize*, sendo eles 0, 1 e 3. Este valor define a quantidade de dados que são utilizados, antes e depois da observação actual para prever a sua classificação [57]. Quer isto dizer que, quando *windowSize* = 0, apenas se assume a observação actual para a geração do modelo de previsão. Quando o valor da janela é superior a 0, consideram-se as x observações antes e as x observações seguintes à observação actual que está a ser analisada, sendo efectuada uma concatenação dos dados.

4.3.2 Resultados e discussão do obtido em CRF_algorithm.m

Os resultados obtidos com este algoritmo encontram-se representados na Tabela 4.2. Para facilitar a sua visualização, gerou-se o gráfico da Figura 4.2, que analogamente ao exibido anteriormente, tem no eixo das abcissas os ficheiros de dados com os vários intervalos de tempo e no eixo das ordenadas as médias das *accuracies*.

Tabela 4.2: Resultados das *accuracies* obtidas com CRF_algorithm.m, para os vários ficheiros indicados e com diferentes valores de *windowSize*.

	Intervalos de tempo		***windowSize* = 0**		***windowSize* = 1**		***windowSize* = 3**	
	Antes (ms)	**Depois (ms)**	**Média da *accuracy***	**Desvio padrão da *accuracy***	**Média da *accuracy***	**Desvio padrão da *accuracy***	**Média da *accuracy***	**Desvio padrão da *accuracy***
Parte 1	200	200	0.1814	0.0195	0.1853	0.0118	0.1934	0.0168
	250	250	0.1883	0.0196	0.1829	0.0143	0.1921	0.0201
	300	300	0.1882	0.0111	0.1803	0.0225	0.2083	0.0372
	350	350	0.1837	0.0179	0.1886	0.0151	0.1968	0.0098
	500	500	0.2001	0.0250	0.1946	0.0412	0.1928	0.0103
	1000	1000	0.2070	0.0237	0.1915	0.0235	0.2089	0.0133
	1500	1500	0.2059	0.0161	0.2126	0.0293	0.2242	0.0309
	2000	2000	0.2047	0.0029	0.1849	0.0195	0.1854	0.0221
Parte 2	200	0	0.1876	0.0154	0.2117	0.0275	0.1937	0.0233
	250	0	0.2011	0.0295	0.2177	0.0269	0.2016	0.0168
	300	0	0.2005	0.0198	0.2085	0.0419	0.1927	0.0341
	350	0	0.2004	0.0281	0.2150	0.0334	0.1983	0.0415
	500	0	0.1961	0.0262	0.2031	0.0604	0.2022	0.0477
	1000	0	0.2053	0.0233	0.2168	0.0285	0.2075	0.0121
	1500	0	0.2076	0.0244	0.2189	0.0255	0.1867	0.0077
	2000	0	0.2133	0.0317	0.2259	0.0295	0.2126	0.0019
Parte 3	500	300	0.1841	0.0173	0.1822	0.0213	0.2059	0.0259
	750	300	0.1952	0.0100	0.1802	0.0274	0.1869	0.0312
	1000	300	0.1920	0.0119	0.2124	0.0177	0.1802	0.0211
	1500	300	0.1962	0.0145	0.2116	0.0286	0.2070	0.0083

Contudo, no gráfico da Figura 4.2, optou-se por representar as *accuracies* através de barras, e não de pontos, tal como apresentado no algoritmo kNN, devido à quantidade significativa de dados para exibir. Esta decisão teve por base ajudar na interpretação e leitura dos resultados, não só referentes ao conjunto de teste, como também os do conjunto de treino, que serão posteriormente exibidos. Além disso, esta representação gráfica também possui os desvios padrão das médias das *accuracies* e as duas barras amarelas verticais, a tracejado, que indicam a divisão entre as 3 partes estipuladas.

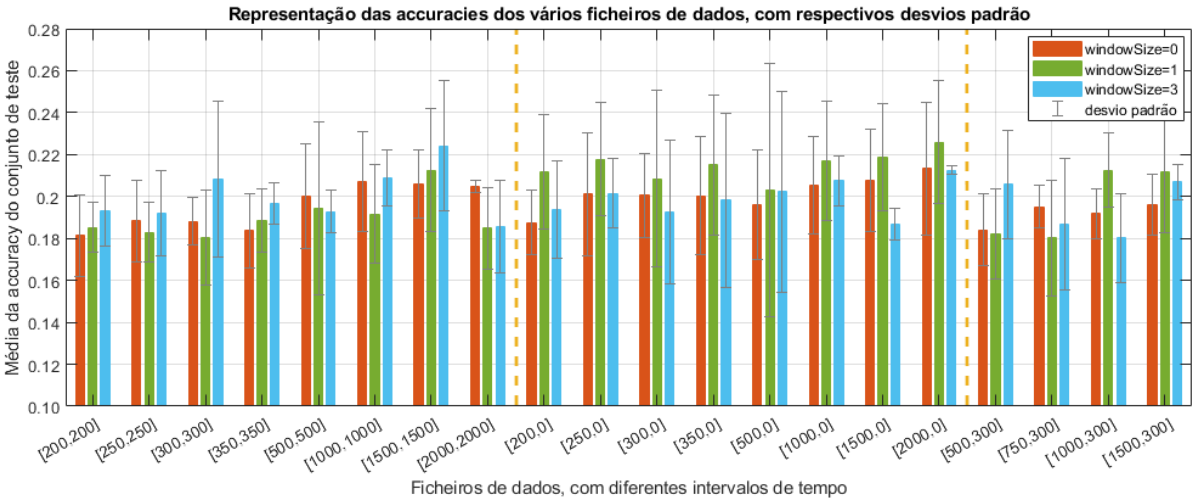


Figura 4.2: Representação gráfica dos valores das *accuracies*, e respectivos desvios padrão, obtidos com o algoritmo CRF, para o conjunto de teste.

Numa primeira instância, verifica-se que, na Parte 2, as *accuracies* mais elevadas foram sempre obtidas com o *windowSize* = 1. Todavia, estes resultados apresentam um desvio padrão superior aos restantes, obtidos com o mesmo ficheiro e com *windowSize* diferentes. A Parte 1 revela que, de um

modo geral, as *accuracies* mais altas foram conseguidas com o *windowSize* = 3 e na terceira parte não se destaca nenhum dos valores deste parâmetro, o que pode dever-se aos poucos ficheiros testados.

Ao analisar o crescimento global do gráfico, é possível observar que as *accuracies* tendem a aumentar quando os intervalos anterior e/ou posterior ao instante do *feedback* aumentam de valor. Comparando os 3 valores de *windowSize*, verifica-se que, em 50% dos casos, as *accuracies* mais altas foram conseguidas com o *windowSize* = 1, seguido do *windowSize* = 3, com os melhores resultados em 35% dos casos.

Os intervalos de tempo que conseguiram as melhores *accuracies*, em cada uma das partes, foram os dos ficheiros [1500,1500]ms com o *windowSize* = 1 e com o *windowSize* = 3 na Parte 1, [1500,0]ms e [2000,0]ms com o *windowSize* = 1 na Parte 2 e [1000,300]ms e [1500,300]ms na terceira parte, também com o *windowSize* = 1. Note-se que, neste conjunto especificado, o parâmetro *windowSize* com o valor 1 é predominante. Embora seja possível observar este facto, verifica-se também a existência de *overfitting* quando o *windowSize* é superior a 1. Neste sentido, foi elaborado o gráfico da Figura 4.3, que mostra as *accuracies* obtidas pelo conjunto de teste (já observadas na Figura 4.2) juntamente com as *accuracies* alcançadas pelo conjunto de treino. Para não dificultar a leitura do gráfico, optou-se por não se apresentar os valores de desvio padrão.

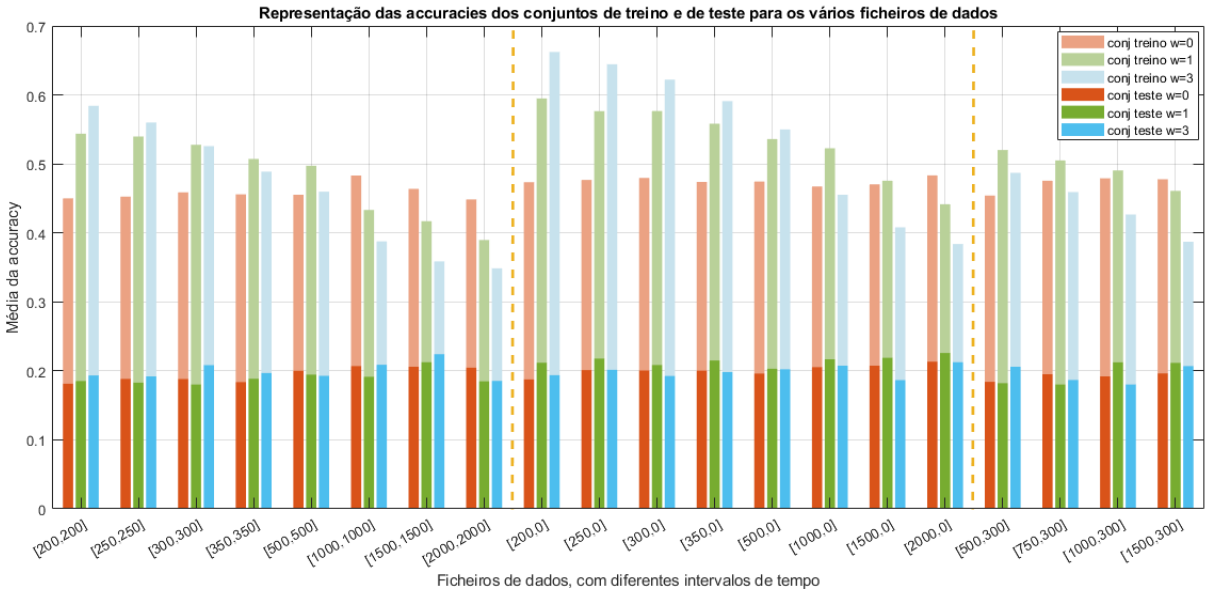


Figura 4.3: Representação gráfica dos valores das *accuracies* obtidos com o algoritmo CRF, para os conjuntos de teste e de treino.

Portanto, o *overfitting* pode ser detectado quando existe um aumento da *accuracy* do conjunto de treino e uma diminuição da *accuracy* do conjunto de teste, com o aumento do *windowSize*. Mais concretamente, se a *accuracy* do conjunto de treino aumenta, espera-se também o aumento da *accuracy* do conjunto de teste. Quando isso não se sucede, pode ser devido à sobreposição dos dados de *labels* diferentes (*overfitting*), que acaba por confundir o classificador. Pela análise da Figura 4.3, é possível constatar que nos primeiros 5 ficheiros da Parte 2 verifica-se a ocorrência de *overfitting* quando o *windowSize* = 3. Tal pode conduzir à ideia de que o ficheiro não tem dados suficientes, isto é, os intervalos

de tempo que abrangem são reduzidos comparativamente aos restantes, pelo que não se deve usar um valor de *windowSize* que englobe muitos dados. Desta forma, quando o ficheiro atinge um certo tamanho, deixa de se observar *overfitting*, quando o *windowSize* toma o valor 3. Nas outras 2 partes, esta conclusão não se aplica, pelo que, apesar das *accuracies* do conjunto de teste terem baixado quando as do conjunto de treino subiram, relativamente ao valor de *windowSize* anterior, poderá não se tratar de *overfitting*.

Além desta análise da *accuracy* geral de cada um dos ficheiros, é também importante examinar, com mais pormenor, as *accuraccies* que foram obtidas para cada uma das *labels* individualmente. Por este motivo, apresentam-se as Tabelas B.2, B.3 e B.4 no Anexo B, onde é possível observar a *accuracy* de cada uma das classes ou, por outras palavras, a "quantidade" de vezes que o classificador etiquetou com a *label* correcta. Juntamente com estas *accuracies*, também se apresentam os respectivos desvios padrão. Para facilitar a visualização e análise destes resultados, foram geradas matrizes de confusão, que mostram não só as *accuracies* de cada *label*, como também o seu erro, ou seja, quando se classificou com a *label* errada. Por exemplo, averiguou-se a "quantidade" de vezes que o classificador etiquetou com a *label* 2, quando a correcta era a *label* 1, ou etiquetou com a *label* 3, quando a certa seria a 1, e por aí adiante.

Do conjunto dos ficheiros que apresentaram as médias das *accuracies* mais elevadas, foram seleccionados um por cada uma das partes, para estudar as suas matrizes de confusão. Assim sendo, escolheram-se ficheiros que tivessem algum dos seus intervalos de tempo em comum e com o mesmo parâmetro de *windowSize*, pelo que se optou por aqueles que continham os intervalos [1500,1500]ms, [1500,0]ms e [1500,300]ms com *windowSize* = 1. As suas matrizes de confusão apresentam-se nas Figuras 4.4, 4.5 e 4.6, respectivamente. O ideal para se obter na matriz de confusão seria ter a média das *accuracies* maior na diagonal principal, pois significaria que o classificador tinha acertado mais vezes na *label* correcta, do que noutra *label* qualquer incorrecta. Contudo, isso não foi o que se observou nas matrizes de confusão obtidas. Analisando estes 3 exemplos, em nenhum se destaca a diagonal principal.

Na Figura 4.4 verifica-se que a maioria dos dados do ficheiro [1500,1500]ms foram classificados como sendo *label* 6, pois é na coluna da classe 6 que se revelam os valores mais elevados. Contudo, a maior parte da classificação feita com esta *label* foi incorrecta, sendo que se etiquetaram mais vezes todas as outras *labels* com a *label* 6 do que a própria. Em sentido contrário, a previsão das *labels* 3, 5 e 7 foi bastante reduzida na sua generalidade, ou seja, houve poucos dados classificados como estas *labels*. Analisando a diagonal principal, nota-se que o classificador conseguiu acertar na etiquetagem da *label* 4, com uma *accuracy* de 0.3840 (ver Tabela B.3). Imediatamente a seguir, encontram-se as *labels* 1 e 6, com *accuracys* de 0.2390 e 0.2458. As restantes apresentam valores de *accuracies* relativamente baixos. Salienta-se ainda que a *accuracy* da *label* 3 é baixa comparativamente às outras classificações feitas com esta mesma *label*. Quer isto dizer que foram classificados mais vezes dados com a *label* 3, sendo, na realidade, *label* 2, do que propriamente os dados da *label* 3. O mesmo aconteceu com a *label* 6, já referido anteriormente, e com a *label* 2. Isto pode ser devido ao facto de haver poucas amostras classificadas com estas *labels*. Recordando a Tabela 3.3 na secção 3.7,

verifica-se que existem poucos exemplos das *labels* 3 e 5, em comparação com as restantes, o que pode justificar a sua baixa *accuracy*. Contudo, tal não é aplicável às *labels* 2 e 6.

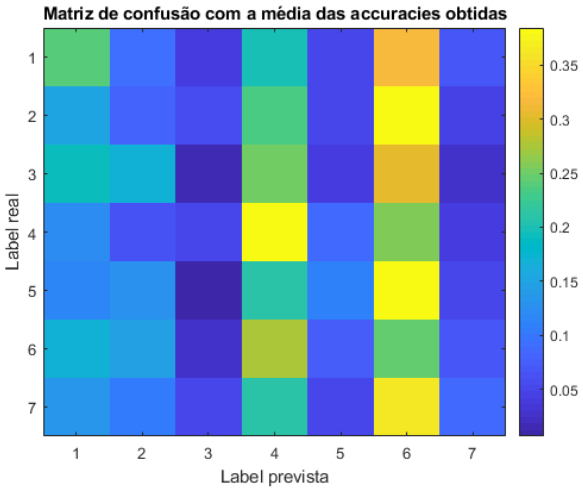


Figura 4.4: Matriz de confusão das *labels*, com o ficheiro [1500,1500]ms e com *windowSize* = 1.

Relativamente aos resultados obtidos com o ficheiro [1500,0]ms, representados na Figura 4.5, observa-se que a *label* 4 continuou com a *accuracy* mais elevada, tendo sido correctamente classificada mais vezes do que as restantes. Desta vez, a *label* 3 apresenta o seu valor de *accuracy* mais elevado na diagonal principal, o que revela uma melhoria, relativamente ao caso anterior, na Figura 4.4. Já não houve tantas classificações incorrectas feitas com a *label* 6, o que também se traduz numa melhoria face ao caso anterior. As *accuraries* das *labels* 1, 5 e 7 são superiores às obtidas com o ficheiro [1500,1500]ms, porém a previsão da *label* 2 continua a ser mais vezes incorrecta do que correcta.

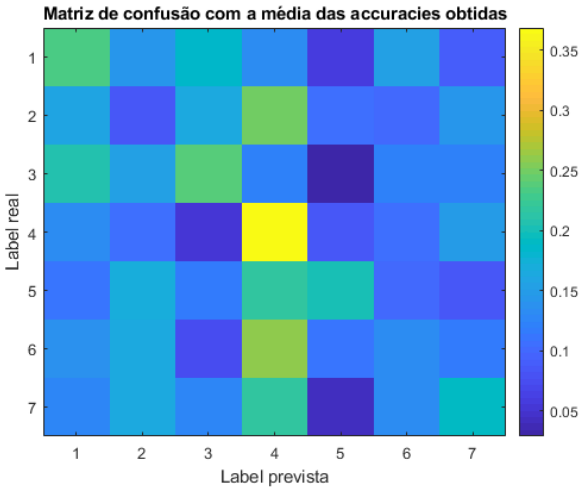


Figura 4.5: Matriz de confusão das *labels*, com o ficheiro [1500,0]ms e com *windowSize* = 1.

No caso dos resultados obtidos com o ficheiro [1500,300]ms, indicados na Figura 4.6, a coluna da *label* 4 mostra que esta foi a mais usada na classificação dos dados e a *accuracy* maior encontra-se na previsão correcta desta mesma *label*. A classificação com as *labels* 2, 6 e 7 continua a ser mais vezes incorrecta do que correcta, o que demonstra piores resultados comparando com o caso da Figura 4.5.

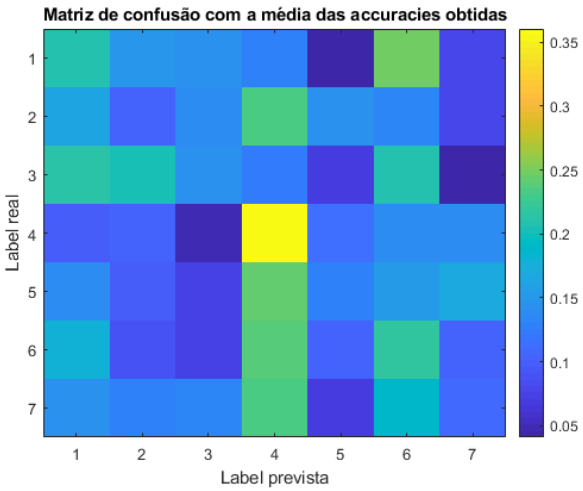
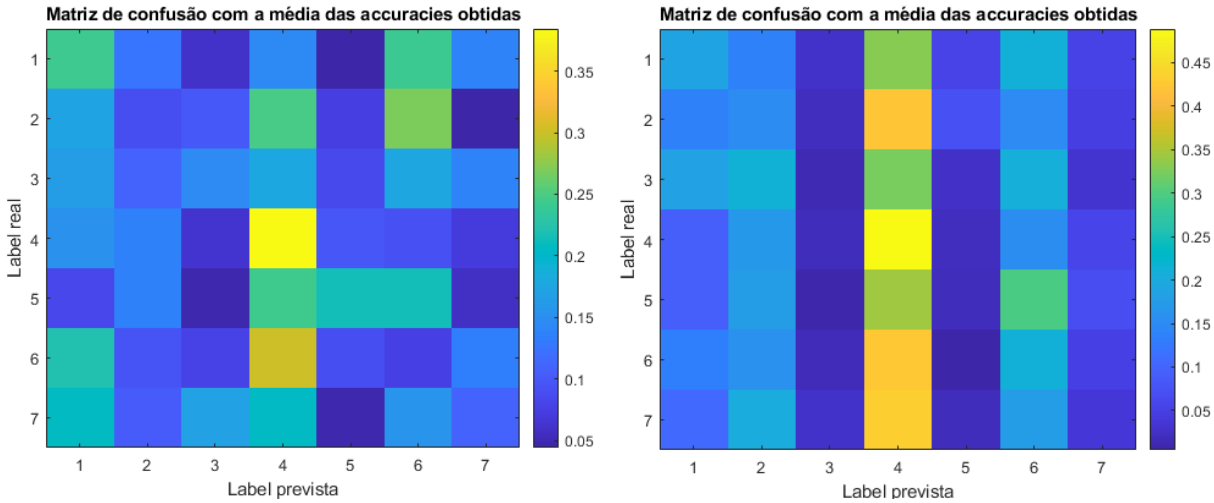


Figura 4.6: Matriz de confusão das *labels*, com o ficheiro [1500,300]ms e com *windowSize* = 1.

Estas matrizes de confusão mostram que a classificação dos dados, principalmente as reveladas nas Figuras 4.5 e 4.6, encontram-se espalhadas pelas 7 *labels* existentes. Apenas se verifica a excepção da *label* 4 que se relevou a predilecta do classificador na etiquetagem das *labels* 2, 4, 5, 6 e 7. Não existe um destaque das *accuracies* na diagonal principal, sendo que o classificador opta mais vezes por *labels* que não são as correctas. Dos 3 ficheiros analisados, aquele que obteve um melhor desempenho foi o [1500,0]ms, pelas razões já identificadas.

Analisando agora as matrizes de confusão de um ficheiro com diferentes valores de *windowSize*, obtêm-se as Figuras 4.7(a) e 4.7(b) para o caso do ficheiro [1500,1500]ms.



(a) Matriz de confusão com *windowSize* = 0.

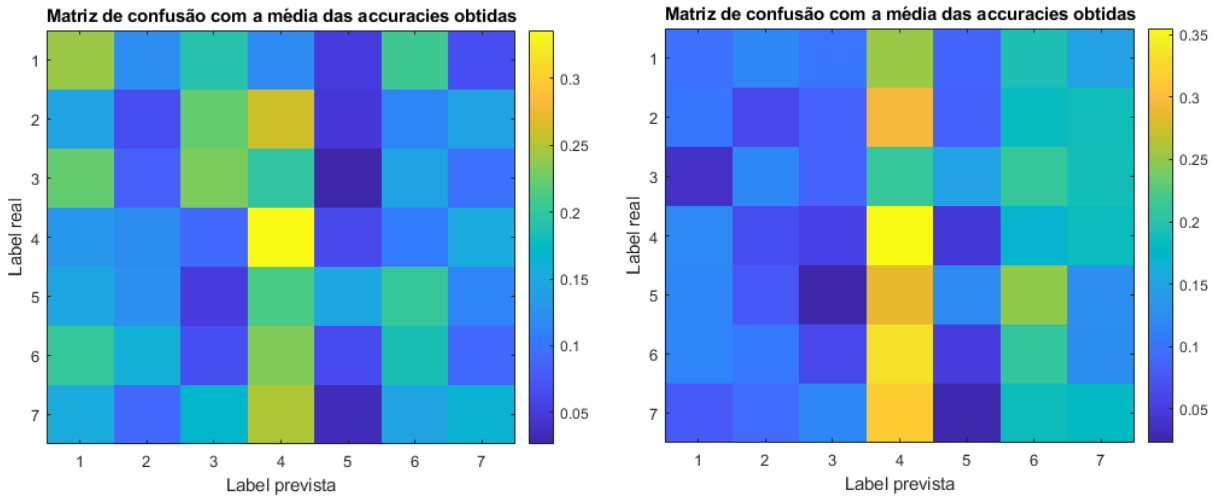
(b) Matriz de confusão com *windowSize* = 3.

Figura 4.7: Matrizes de confusão das *labels* do ficheiro [1500,1500]ms, para diferentes valores de *windowSize*.

É possível verificar que, com o aumento do valor do parâmetro *windowSize*, a classificação com certas *labels* torna-se mais notória. Por exemplo, com o *windowSize* = 3, houve poucos dados classificados com as *labels* 3 e 5, ao contrário da *label* 4 que foi bastante utilizada nas previsões. Tal pode dever-se à quantidade de dados existentes para treinar o classificador, pois há mais exemplos da *label*

4 e relativamente poucos das *labels* 3 e 5 (rever Tabela 3.3). A *accuracy* geral é mais elevada com o *windowSize* = 3, apesar das suas previsões tendenciosas. O desempenho com o parâmetro *windowSize* = 1 pode ter sido influenciado pela preferência notória da *label* 6 nas suas previsões, observado na Figura 4.4, que estava incorrecta, na grande maioria das vezes. Em nenhum dos 3 casos sobressai a diagonal principal.

Posto isto, ir-se-à analisar outro conjunto de resultados, com o mesmo ficheiro e diferentes valores de *windowSize*. Foi escolhido o ficheiro [1500,0]ms para dar seguimento ao analisado anteriormente. Apresentam-se, portanto, as matrizes de confusão das *labels* com o *windowSize* = 0 e com o *windowSize* = 3 nas Figuras 4.8(a) e 4.8(b), respectivamente.



(a) Matriz de confusão com *windowSize* = 0.

(b) Matriz de confusão com *windowSize* = 3.

Figura 4.8: Matrizes de confusão das *labels* do ficheiro [1500,0]ms, para diferentes valores de *windowSize*.

Analisando estas matrizes juntamente com a apresentada na Figura 4.5, observa-se que, quando o *windowSize* = 3, existe uma preferência notória para a etiquetagem com a *label* 4, não estando totalmente correcta, pois há ainda algumas classificações previstas como *label* 4 quando na realidade são as *labels* 2, 5, 6 e 7. Relativamente à diagonal principal, revela-se que no caso do *windowSize* = 0, obtêm-se piores resultados somente na *label* 2, ou seja, existem *labels* classificadas mais vezes como sendo *label* 2 (incorrectamente), do que a própria. O mesmo não se verifica na diagonal principal do *windowSize* = 3, que mostra uma distribuição uniforme na previsão das *labels* 1, 3, 6 e 7 e, novamente pior, na *label* 2, o que conduz à sua baixa *accuracy* geral comparativamente às outras com o mesmo ficheiro. Com isto, é possível constatar que, quando o valor do *windowSize* é 0 ou 1, obtêm-se melhores previsões do que quando esse parâmetro é igual a 3, uma vez que a previsão da classificação com uma determinada *label* é mais vezes correcta do que incorrecta.

Com o objectivo de melhorar os resultados das *accuracies*, optou-se por experimentar e testar o algoritmo HCRF, o que nos leva à próxima secção.

4.4 Algoritmo HCRF

Este algoritmo foi aplicado em duas situações distintas, implementadas nos ficheiros HCRF_algorithm.m e HCRF_algorithm_final.m. A sua diferença consiste na replicação dos dados dos conjuntos de treino, executada pelo último ficheiro de código indicado. Esta replicação pretende melhorar o treino do classificador e, consequentemente, obter melhores resultados do que aqueles que foram alcançados no primeiro caso (sem replicação).

Neste algoritmo, definem-se 2 parâmetros que irão influenciar os resultados finais, sendo eles o *nbHiddenStates* e o *windowSize*. Por este motivo, foram estudados vários valores que permitissem detectar qual a combinação onde se atingiam os melhores valores de *accuracy*. Tal como já tinha sido referido na secção 3.6.3, o *nbHiddenStates* refere-se ao número de estados escondidos que o algoritmo vai assumir, considerando a existência de dependências entre eles. O parâmetro *windowSize* foi anteriormente explicado na secção 4.3.1.

4.4.1 Testes com o ficheiro HCRF_algorithm.m

Para esta análise, foram utilizados os ficheiros [300,300]ms e [500,300]ms, pois considera-se importante ter em atenção o erro, dos 300 milissegundos, que pôde existir no alinhamento dos vídeos, tal como foi explicado na secção 4.1. Por este motivo, o intervalo posterior ao instante do *feedback* considerado foi de 300 milissegundos. Relativamente ao intervalo de tempo que antecede esse instante, optou-se também pelos 300 milissegundos e um valor superior, de 500 milissegundos. A gama de valores usada para testar os parâmetros *nbHiddenStates* e *windowSize*, com estes ficheiros, foi 1, 2, 3 e 5 para o primeiro e 1, 3 e 5 para o segundo. Pretende-se, deste modo, encontrar um padrão que permita tirar conclusões sobre os mesmos.

4.4.2 Resultados e discussão do obtido em HCRF_algorithm.m

Ficheiro [300,300]ms

Portanto, com o ficheiro [300,300]ms e com as várias combinações dos parâmetros, indicadas anteriormente, obtiveram-se os resultados para as médias das *accuracies* e respectivos desvios padrão presentes na Tabela 4.3. Estes resultados dizem respeito ao teste do classificador efectuado com o conjunto de teste. À primeira vista, verifica-se que os valores das *accuracies* são bastante idênticos, oscilando num intervalo de 0.205 a 0.253. Relativamente aos seus desvios padrão, estes variam entre os 0.016 e 0.06, atingindo até valores relativamente altos, tendo em consideração os resultados obtidos nas médias.

Para facilitar a visualização destes resultados, gerou-se o gráfico apresentado na Figura 4.9, que conjuga os valores das médias das *accuracies* obtidas com os seus desvios padrão. Foram também realizados testes com os conjuntos de treino, utilizados para gerar os classificadores, apenas com o objectivo de estudar a existência de *overfitting*. Posto isto, na Figura 4.9 também estão indicadas as *accuracies* dos conjuntos de treino, com os respectivos desvios padrão.

Tabela 4.3: Resultados da *accuracy* com o código HCRF_algorithm.m, para o ficheiro [300,300]ms.

Parâmetros		**Resultados - Ficheiro [300,300]ms**	
nbHiddenStates	***windowSize***	**Média da *accuracy***	**Desvio padrão da *accuracy***
1	1	0.2529	0.0240
	3	0.2529	0.0240
	5	0.2529	0.0240
2	1	0.2225	0.0267
	3	0.2294	0.0379
	5	0.2271	0.0326
3	1	0.2306	0.0389
	3	0.2494	0.0167
	5	0.2294	0.0414
5	1	0.2319	0.0364
	3	0.2095	0.0402
	5	0.2059	0.0594

Tal como anteriormente, as barras verticais, associadas a cada ponto, medem o desvio padrão dessa *accuracy* e as 3 barras amarelas, a tracejado, correspondem a linhas auxiliares, que separam, por grupos, os resultados obtidos, de acordo com o seu valor de *nbHiddenStates*. Para simplificar a representação, abreviou-se o nome de *windowSize* para *w* e de *nbHiddenStates* para *h*.

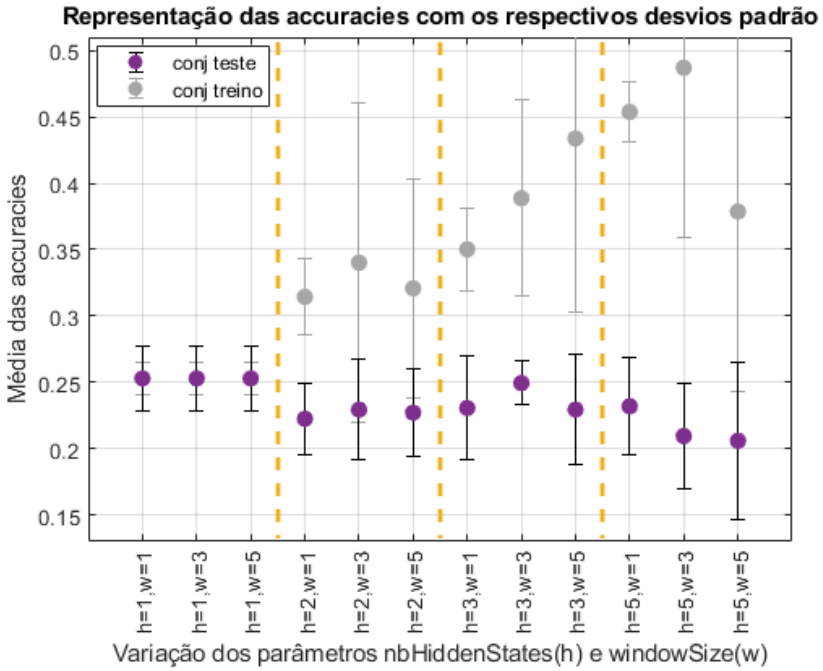


Figura 4.9: Representação gráfica das *accuracies* obtidas com o código HCRF_algorithm.m, para o ficheiro [300,300]ms.

Numa primeira instância, é possível observar que, quando o *nbHiddenStates* = 1, as médias das *accuracies* têm sempre o mesmo valor, apesar da variação do *windowSize*. Isto verifica-se para ambos os conjuntos de treino e de teste. Além disso, os desvios padrão de cada um dos conjuntos também

se mantêm constantes com a variação do *windowSize*. Este acontecimento será analisado mais pormenorizadamente na secção 4.5.1. Relativamente ao conjunto de teste e para os outros valores do *nbHiddenStates*, observa-se a existência de um decréscimo da *accuracy*, quando se passa do *windowSize* = 3 para o *windowSize* = 5. Quando o parâmetro *nbHiddenStates* toma os valores 2 e 3, verifica-se uma subida da *accuracy* ao aumentar o *windowSize* de 1 para 3. O mesmo não se aplica para o *nbHiddenStates* = 5.

O parâmetro *windowSize* está directamente relacionado com a ocorrência de *overfitting*, pois quando este parâmetro ultrapassa um certo valor, dá-se o seu aparecimento. O *overfitting* corresponde a um ajuste muito específico do modelo ao conjunto de dados fornecido, prejudicando, assim, a previsão de novos resultados para um outro conjunto de dados. Além disso, se estes tiverem ruído, é feita uma adaptação bastante complexa, para se conseguir modelar esse "erro", conduzindo a modelos imprecisos. Neste caso, é possível observar a ocorrência de *overfitting* quando se verifica um aumento da *accuracy* no conjunto de treino, utilizado na geração do modelo, juntamente com uma diminuição da *accuracy* do conjunto de teste. Deste modo e pela análise da Figura 4.9, verifica-se a ocorrência de *overfitting* quando o *nbHiddenStates* = 3 e o *windowSize* = 5, e quando o *nbHiddenStates* = 5 e o *windowSize* = 3, uma vez que se observa um aumento da *accuracy* do conjunto de treino e um decréscimo da *accuracy* do conjunto de teste, face aos casos anteriores com um *windowSize* inferior. A partir do momento que existe *overfitting* com um determinado valor de *windowSize*, existirá também para outros valores de *windowSize* superiores, mantendo os restantes parâmetros constantes.

No que diz respeito ao desvio padrão apresentado pelo conjunto de teste, cada grupo de *nbHiddenStates* iguais revela um desvio padrão com amplitudes semelhantes. A única excepção, que sobressai bastante das restantes, encontra-se no grupo com *nbHiddenStates* = 3. Neste caso, quando o parâmetro *windowSize* também toma o valor 3, o desvio padrão é muito inferior relativamente aos outros dois do mesmo grupo.

As *accuracies* mais elevadas foram obtidas com os conjuntos de parâmetros *nbHiddenStates* = 3 e *windowSize* = 3 e com *nbHiddenStates* = 5 e *windowSize* = 1.

Para que se consigam retirar mais conclusões destes resultados, é necessário examinar também a *accuracy* de cada uma das *labels*, obtida com o conjunto de teste. Estas são disponibilizadas na Tabela B.5 do Anexo B, juntamente com os seus valores de desvio padrão. À primeira vista, sobressai a quantidade de zeros que se apresentam nas médias das *accuracies*. Isto significa que houve *labels* que foram previstas sempre de forma incorrecta na classificação dos dados.

Com o *nbHiddenStates* a tomar o valor 1, os resultados são novamente todos iguais, como seria de esperar. Verifica-se agora que os dados foram somente classificados com as *labels* 1 e 4. Mais especificamente, o classificador conseguiu acertar em todos os dados classificados com a *label* 4. No entanto, isto seria uma boa notícia, caso 99.65% das observações não tivessem sido classificadas com esta *label*. Portanto, a *accuracy* obtida com o *nbHiddenStates* = 1 é 0.25 aproximadamente (ver Tabela 4.3), pois o número de exemplos etiquetados com a *label* 4 constitui cerca de 24% do total de amostras. Sendo que se acertou em todas as *labels* 4 e numa pequena percentagem de *labels* 1, isso perfaz o valor da *accuracy* geral de 0.25 atingido.

É também possível notar que, mantendo o *windowSize* constante e aumentando o *nbHiddenStates*, o número de zeros nas *accuracies* vai diminuindo. Por exemplo, para o *windowSize* = 1, existem 5 zeros quando o *nbHiddenStates* = 1; 3 zeros com o *nbHiddenStates* = 2; 2 zeros para o *nbHiddenStates* = 3; e nenhum zero, quando se aumenta o número de estados escondidos para 5. Isto transmite a ideia de que, para a mesma janela de dados a utilizar na previsão, são classificadas correctamente mais *labels* com o aumento do número de estados escondidos. O mesmo se verifica quando se fixa o *windowSize* com os valores 3 e 5. Relativamente ao caso do *windowSize* = 5, parece ser necessário aumentar ainda mais o número de estados escondidos, de forma a reduzir o número de zeros nas *accuracies* e, consequentemente, etiquetar mais *labels* correctamente.

Na análise da Tabela B.5, foi possível encontrar uma situação em que o classificador começa a discriminar certas *labels* na sua previsão. Mantendo o mesmo número de estados escondidos e aumentando o *windowSize*, existe, por vezes, um aumento do número de zeros nas *accuracies* das *labels*, o que significa que se deixaram de utilizar certas *labels* na previsão do modelo. Tal pode estar associado à presença de *overfitting* nos resultados. Contudo, nem sempre existe *overfitting* quando se detecta esta situação. Por exemplo, quando o *nbHiddenStates* = 2 e se aumenta o *windowSize* de 3 para 5, verifica-se um decréscimo nas classes utilizadas na previsão, pois deixou de se usar as *labels* 3 e 6. Neste caso, não se verifica a existência de *overfitting*. Outro exemplo encontra-se quando *nbHiddenStates* = 3, pois existe um aumento dos zeros nas *accuracies* das *labels* com o aumento do *windowSize*. Apenas num destes aumentos, não se verifica a ocorrência de *overfitting*.

Intuitivamente, classificar correctamente mais *labels* deveria corresponder à melhoria da *accuracy* geral do classificador, mas isso nem sempre acontece. Tal é visível quando se estipula o *windowSize* com o valor 3. Embora o número de zeros vá diminuindo com o aumento do *nbHiddenStates*, a *accuracy* geral do classificador diminui quando se passa do *nbHiddenStates* = 3 para *nbHiddenStates* = 5.

É, por isso, importante combinar estes dois parâmetros, *nbHiddenStates* e *windowSize*, de modo a que se consigam classificar correctamente o maior número de *labels* possíveis, aumentando, simultaneamente, a *accuracy* geral do classificador. Isto foi conseguido para o *windowSize* = 1, que revelou um aumento da *accuracy* geral à medida que se aumentava o parâmetro *nbHiddenStates* e se tinham em consideração mais *labels* na previsão.

De seguida, são exibidas várias matrizes de confusão que permitem visualizar a diferença das *accuracies* dentro de cada classe. Para isso, fixou-se o parâmetro *windowSize* e variou-se o *nbHiddenStates*, uma vez que é quando se verifica uma maior alteração nos valores das *accuracies* gerais. Foi, então, seleccionado o *windowSize* com o valor 3, pelo mesmo motivo.

Na Figura 4.10, verifica-se que o classificador previu quase todos os dados como sendo da *label* 4, tal como já tinha sido referido anteriormente. A única excepção encontra-se na *label* 1, que revela uma *accuracy* diferente de zero. Contudo, esta é pouco perceptível na Figura 4.10, pois corresponde a um valor muito próximo de zero.

Relativamente à Figura 4.11, é evidente a existência de classificações com mais *labels* do que no caso anterior (Figura 4.10). Porém, a *label* 4 continua a ser eleita do classificador. É ainda importante notar que não houve nenhuma classificação realizada com as *labels* 5 e 7, nem mesmo incorrectas.

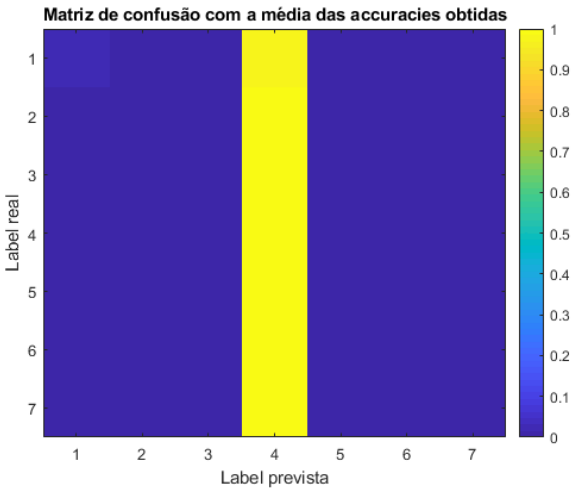


Figura 4.10: Matriz de confusão com as *accuracies* do ficheiro [300,300]ms para $h = 1$ e $w = 3$.

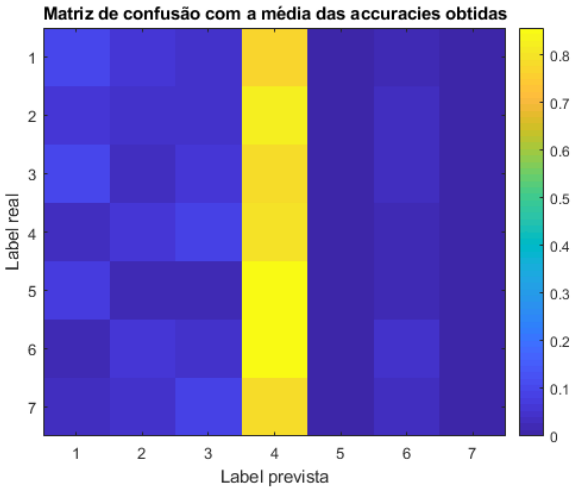


Figura 4.11: Matriz de confusão com as *accuracies* do ficheiro [300,300]ms para $h = 2$ e $w = 3$.

Por outro lado, na Figura 4.12 não houve nenhuma classificação feita com as *labels* 3 e 5. Este facto pode dever-se à existência de poucos dados classificados com estas duas *labels*, relativamente às restantes, como observado na Tabela 3.3 da secção 3.7. Contudo, esta justificação não se aplica para a *label* 7, referida no caso anterior (Figura 4.11), pois existem bastantes dados classificados com essa etiqueta. Nesta matriz de confusão, verifica-se, novamente, uma redução da incidência na classificação dos dados com a *label* 4 e um crescimento da classificação com as *labels* 1, 2 e 6, mesmo sendo na sua maioria classificações incorrectas. Além disso, salienta-se a maior utilização da *label* 6 nas previsões da classificação dos dados, comparativamente aos exemplos das Figuras 4.10 e 4.11.

Por último, a matriz de confusão apresentada na Figura 4.13, demonstra novamente que a *label* 4 continua a ser a mais aplicada na classificação dos dados. No entanto, é também visível que as restantes *labels* começam a ser mais utilizadas. Houve um aumento considerável na classificação com a *label* 7, apesar da maioria dessas classificações estar incorrecta. Neste caso, todas as *labels* foram classificadas correctamente, em algum momento da previsão, ou seja, já não se encontram nenhuns valores a zero nas *accuracies* indicadas na Tabela B.5.

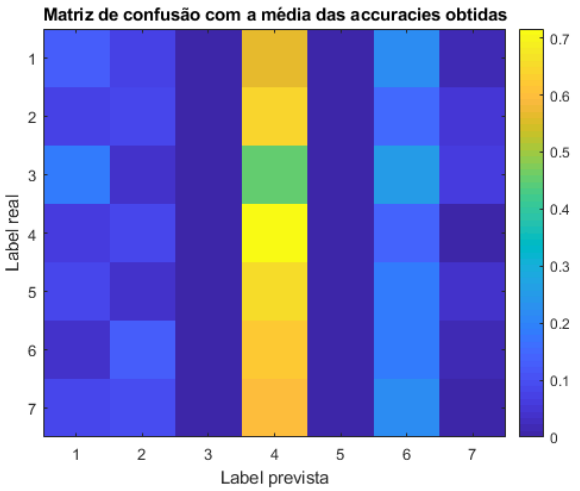


Figura 4.12: Matriz de confusão com as *accuracies* do ficheiro [300,300]ms para $h = 3$ e $w = 3$.

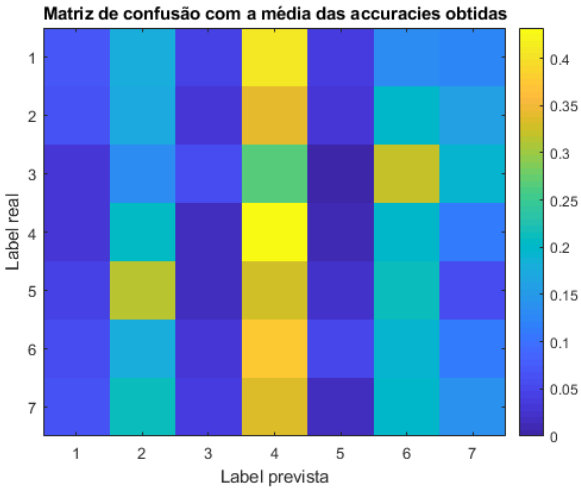


Figura 4.13: Matriz de confusão com as *accuracies* do ficheiro [300,300]ms para $h = 5$ e $w = 3$.

Na matriz de confusão da Figura 4.13, é ainda possível fazer a correspondência entre os resultados obtidos e o número de dados etiquetados com cada uma das *labels* (ver Tabela 3.3). Por outras palavras, a quantidade de previsões feitas com cada uma das *labels* é proporcional ao número de dados classificados com cada uma delas. Por exemplo, a *label* 4 é a que contém mais amostras, pelo que é a mais usada nas previsões do classificador. Seguem-se as *labels* 2 e 6, que têm, aproximadamente, o mesmo número de dados e as suas colunas na Figura 4.13 são semelhantes. Logo de seguida, encontra-se a *label* 7, que tem menos amostras do que as 3 *labels* anteriores. A *label* 1 que possui mais amostras que a *label* 7, é o caso excepcional, pois é utilizada menos vezes nas classificações do que a *label* 7. Por último, com poucas classificações, comparativamente às restantes, encontram-se as *labels* 3 e 5, que também são aquelas que possuem menos amostras.

Adicionalmente, é analisada a matriz de confusão obtida com os parâmetros *windowSize* = 1 e *nbHiddenStates* = 5, que também corresponde à combinação de valores onde todas as *labels* foram correctamente utilizadas, pelo menos uma vez. Quer isto dizer que na Tabela B.5, todos os valores das *accuracies* das *labels* são diferentes de zero, tal como no caso anterior (Figura 4.13). Além disso, com

esta combinação de parâmetros, a *accuracy* geral é superior. Na Figura 4.14 apresenta-se, então, a sua matriz de confusão.

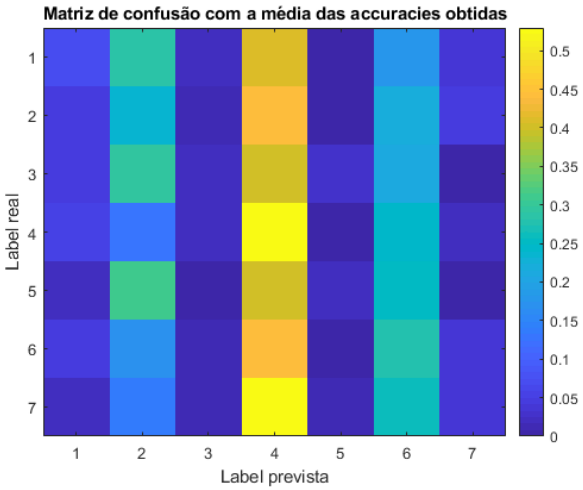


Figura 4.14: Matriz de confusão com as *accuracies* do ficheiro [300,300]ms para $h = 5$ e $w = 1$.

Tal como é possível verificar, esta matriz de confusão é bastante semelhante à anterior, da Figura 4.13. Contudo, nesta destacam-se mais as diferenças entre as previsões de classificação de cada uma das *labels*, ou seja, a "preferência" do classificador para certas *labels* é mais notória. Adicionalmente, aquela excepção que se observou na Figura 4.13 para a *label* 1, que mostrava uma *accuracy* superior à da *label* 7 embora tivesse menos amostras, já não se verifica. Os resultados destas *accuracies* cumprem a proporção do número de amostras de cada *label* (rever Tabela 3.3). É ainda importante notar que, neste caso, verificou-se a existência de *overfitting*.

Ficheiro [500,300]ms

No que diz respeito ao ficheiro [500,300]ms, foram realizados os mesmos testes executados com o ficheiro [300,300]ms. Os resultados obtidos estão presentes na Tabela 4.4 e são relativos ao conjunto de teste.

Numa primeira instância, verifica-se que todos os valores das *accuracies* são superiores a 0.20, oscilando, mais especificamente, num intervalo entre 0.201 e 0.253. Os seus desvios padrão mostram uma variação entre os 0.009 e 0.048, o que demonstra uma melhoria relativamente ao obtido com o ficheiro [300,300]ms. Mais uma vez, observa-se que todos os resultados com o *nbHiddenStates* = 1 são iguais, tanto as médias das *accuracies* como os respectivos desvios padrão. Além disso, estes resultados são exactamente iguais aos obtidos com o ficheiro [300,300]ms.

Para auxiliar na análise dos restantes resultados, criou-se o gráfico apresentado na Figura 4.15, que mostra não só os resultados obtidos com o conjunto de teste, como também os obtidos com o conjunto de treino. Estes últimos resultados serão apenas utilizados para detectar a ocorrência de *overfitting*.

De forma análoga ao adquirido com o ficheiro [300,300]ms, os valores das *accuracies* obtidos para o *nbHiddenStates* = 1 são exactamente iguais, independentemente do valor do *windowSize* e quer seja o conjunto de teste ou o conjunto de treino. Para os outros valores de *nbHiddenStates*, observa-se

Tabela 4.4: Resultados da *accuracy* com o código HCRF_algorithm.m, para o ficheiro [500,300]ms.

Parâmetros		**Resultados - Ficheiro [500,300]ms**	
nbHiddenStates	***windowSize***	**Média da *accuracy***	**Desvio padrão da *accuracy***
1	1	0.2529	0.0240
	3	0.2529	0.0240
	5	0.2529	0.0240
2	1	0.2470	0.0273
	3	0.2236	0.0472
	5	0.2132	0.0418
3	1	0.2109	0.0435
	3	0.2365	0.0346
	5	0.2190	0.0238
5	1	0.2083	0.0346
	3	0.2282	0.0405
	5	0.2014	0.0090

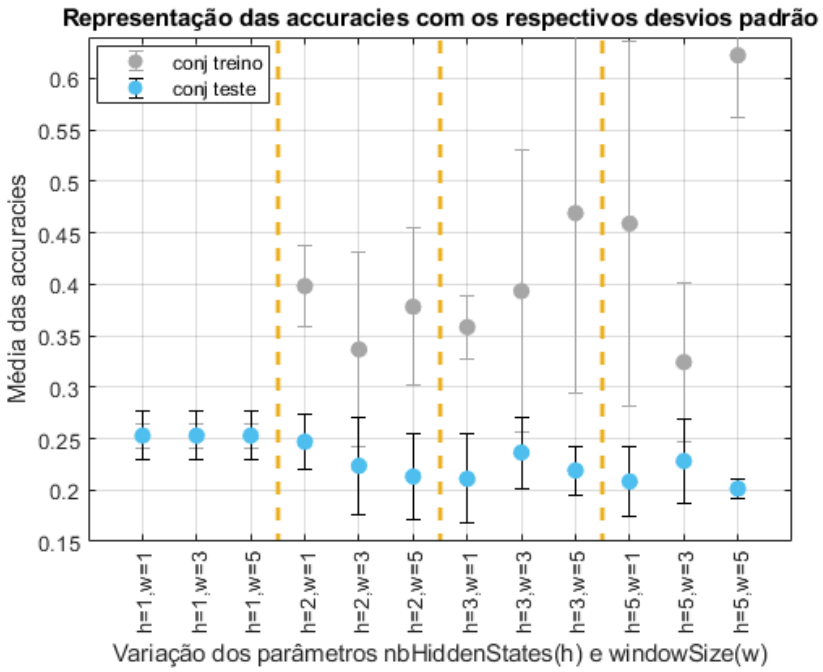


Figura 4.15: Representação gráfica das *accuracies* obtidas com o código HCRF_algorithm.m, para o ficheiro [500,300]ms.

que existe sempre uma diminuição da *accuracy* do conjunto de teste, quando se passa do *windowSize* = 3 para o *windowSize* = 5. Esta diminuição é sempre acompanhada pelo aumento da *accuracy* do conjunto de treino, pelo que é possível concluir que existe *overfitting* quando o *windowSize* = 5, para qualquer valor de *nbHiddenStates* testado.

Os desvios padrão apresentados pelo conjunto de teste são bastante idênticos, apesar da variação dos parâmetros. Nota-se um destaque quando o *nbHiddenStates* = 5 e o *windowSize* = 5, pois este desvio padrão é bastante reduzido e inferior aos restantes.

As *accuracies* mais elevadas foram obtidas com os conjuntos de parâmetros *nbHiddenStates* = 2 e *windowSize* = 1, *nbHiddenStates* = 3 e *windowSize* = 3 e, por último, com *nbHiddenStates* = 5 e *windowSize* = 3.

Para complementar esta análise, é necessário estudar as *accuracies* obtidas por cada uma das *labels* individualmente. Nesse sentido, os seus resultados relativos ao conjunto de teste encontram-se discriminados na Tabela B.6 do Anexo B. Quando o número de estados escondidos é 1, os resultados são novamente todos iguais, independentemente da variação do *windowSize*, e são também iguais ao obtido com o ficheiro [300,300]ms. Deste modo, a análise feita anteriormente, a respeito dos valores obtidos pelas *labels* 1 e 4, também se aplica para este caso.

Mantendo o *nbHiddenStates* constante e aumentando o *windowSize*, verifica-se que não é possível estabelecer um padrão de aumento ou diminuição do número de zeros apresentados em cada linha da Tabela B.6. Relembrando, cada zero nesta tabela significa que essa *label* nunca foi prevista correctamente ou, por outras palavras, se essa *accuracy* for diferente de zero significa que se classificou correctamente essa *label* para uma amostra, no mínimo. Neste caso, tanto se observa um aumento do número de zeros como uma diminuição, com a subida do *windowSize*. Por exemplo, quando *nbHiddenStates* = 2 e *windowSize* = 1, tem-se apenas 1 zero, apresentado pela *label* 5. Passando para o *windowSize* = 3, tem-se agora 2 zeros associados às *labels* 3 e 5. Para um *windowSize* superior, de valor 5, não é apresentado nenhum zero. Além disso, mantendo o *windowSize* constante e aumentando o *nbHiddenStates*, também não é possível estabelecer um padrão. Por exemplo, quando *windowSize* = 1, aumenta o número de zeros quando se passa de *nbHiddenStates* = 2 para *nbHiddenStates* = 3. Por outro lado, quando *windowSize* = 3, verifica-se um decréscimo quando se passa de *nbHiddenStates* = 2 para *nbHiddenStates* = 3.

Estes resultados não sustentam as ideias formuladas no caso anterior, para o ficheiro [300,300]ms. Contudo, e de um modo geral, o ficheiro [500,300]ms apresenta uma redução de cerca de um terço do número de zeros, apresentados na Tabela B.6, comparativamente ao revelado na Tabela B.5, ou seja, foram classificadas correctamente mais *labels*. É ainda importante notar que, nos dois casos que revelaram ter classificado correctamente todas as *labels*, em algum momento da previsão, obtém-se das *accuracies* gerais mais baixas, apresentadas na Tabela 4.4, e representam dois dos três casos onde ocorreu *overfitting*.

Portanto, sublinha-se a importância de haver uma boa combinação dos parâmetros *nbHiddenStates* e *windowSize*, para que se atinjam resultados onde se utilizem o maior número de *labels* nas previsões das classificações, simultaneamente com o aumento da *accuracy* geral do modelo.

Ainda assim, analisou-se com mais detalhe a evolução das *accuracies* obtidas dentro de cada classe. Foi, então, seleccionado o parâmetro *nbHiddenStates* = 2, por apresentar uma maior alteração da *accuracy* geral, com a variação do *windowSize*. Nas Figuras 4.16, 4.17 e 4.18 estão disponibilizadas as suas matrizes de confusão.

Na matriz de confusão da Figura 4.16, verifica-se que houve uma tendência para a classificação com as *labels* 2 e 4, com maior incidência na última. A *label* 5, que apresentou um zero na Tabela B.6, revela agora que essa *label* foi de facto utilizada na previsão dos dados, mas sempre incorrectamente.

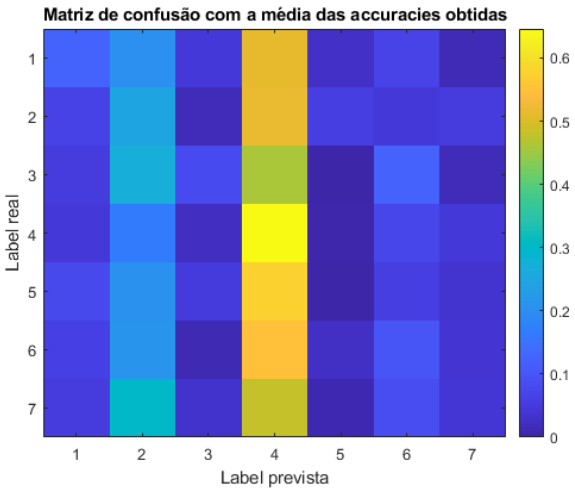


Figura 4.16: Matriz de confusão com as *accuracies* do ficheiro [500,300]ms para $h = 2$ e $w = 1$.

No estudo da Figura 4.17, é possível notar que as *labels* 3 e 5 nunca foram utilizadas nas previsões do classificador. Por outro lado, a *label* 4 foi novamente a mais utilizada. Observa-se ainda que a *label* 2 classificou o mesmo número de vezes dados de todas as classes, o que não se verificou no caso anterior, da Figura 4.16, onde as previsões mostram estar mais distribuídas. Comparando também com a Figura 4.12, onde foram utilizados os mesmos parâmetros de *nbHiddenStates* e *windowSize*, averigua-se que, de um modo geral, o ficheiro [500,300]ms apresenta uma maior distribuição nas suas previsões pelas 7 *labels* existentes. No entanto, tal não contribui para uma melhoria da *accuracy* geral do classificador.

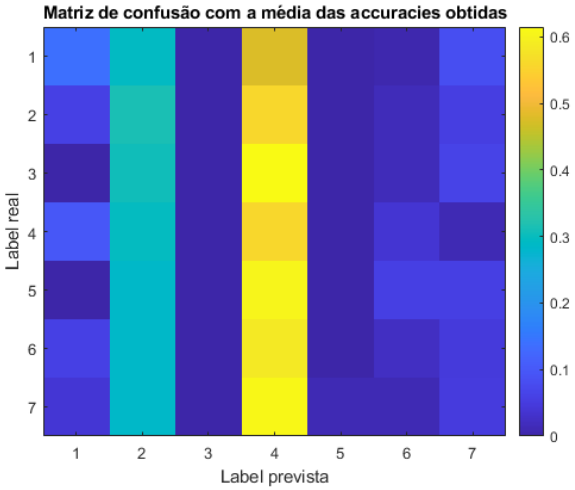


Figura 4.17: Matriz de confusão com as *accuracies* do ficheiro [500,300]ms para $h = 2$ e $w = 3$.

Relativamente à matriz de confusão da Figura 4.18, revela-se que a etiquetagem com a *label* 2 melhorou e que começaram a ser utilizadas as *labels* 3 e 5. A diagonal principal mostra que todas as *labels* foram usadas correctamente em algum momento da previsão, apesar de o classificador escolher mais vezes utilizar uma *label* incorrecta. Por exemplo, a *label* 4 foi prevista para classificar mais vezes os dados que eram na realidade da *label* 5, do que da própria *label* 4. Além disso, este foi um dos

casos onde se observou a existência de *overfitting*.

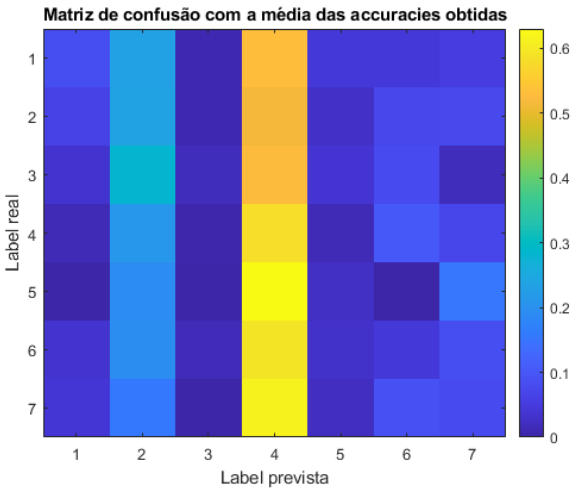


Figura 4.18: Matriz de confusão com as *accuracies* do ficheiro [500,300]ms para $h = 2$ e $w = 5$.

Dadas estas inconsistências nos resultados e da tendência da previsão para as *labels* com um maior número de amostras, ponderou-se equilibrar o número de dados etiquetados, com cada uma das *labels*. Assim, pretende-se evitar esta tendência dos resultados, pelo que se introduz a próxima secção.

4.4.3 Testes com o ficheiro HCRF_algorithm_final.m

Com o objectivo de melhorar o desempenho do classificador, considerou-se a hipótese de uniformizar o conjunto de dados utilizados no seu treino. Por outras palavras, equilibrou-se o número de amostras existentes para cada uma das *labels*. No código MATLAB, após se definir o conjunto de dados a usar no treino, procura-se o número de amostras de cada *label* e fixa-se o seu valor mais elevado. De seguida, procede-se à replicação dos dados das outras 6 *labels*, até perfazerem esse valor máximo. Assim, garante-se que todas as *labels* possuem o mesmo número de amostras, promovendo a eliminação da tendência/preferência para determinadas classes. É de notar que esta replicação foi apenas implementada nos conjuntos de treino, para que não houvesse ainda mais manipulação dos dados.

Para esta análise foram utilizados os mesmos ficheiros de dados usados na secção 4.4.2, que contemplam os intervalos de tempo [300,300]ms e [500,300]ms. Os valores dos parâmetros *nbHiddenStates* e *windowSize* também se mantêm iguais aos estudados anteriormente, no caso do algoritmo HCRF sem replicação.

4.4.4 Resultados e discussão do obtido em HCRF_algorithm_final.m

Ficheiro [300,300]ms

Para o ficheiro com o intervalo de tempo [300,300]ms, obtiveram-se os resultados apresentados na Tabela 4.5, para o estudo efectuado com os conjuntos de teste. Neste caso, os valores das *accuracies* oscilaram mais do que as indicadas na Tabela 4.3. Esta variação engloba valores entre os 0.111 e

os 0.206 de média. Os desvios padrão atingiram valores relativamente elevados, comparando com as respectivas médias das *accuracies*, e abrangem resultados entre os 0.0238 e 0.0783.

Tabela 4.5: Resultados da *accuracy* com o código HCRF_algorithm_final.m, para o ficheiro [300,300]ms.

Parâmetros		**Resultados - Ficheiro [300,300]ms**	
nbHiddenStates	***windowSize***	**Média da *accuracy***	**Desvio padrão da *accuracy***
1	1	0.1989	0.0611
	3	0.2060	0.0399
	5	0.1911	0.0783
2	1	0.1346	0.0603
	3	0.1673	0.0543
	5	0.1359	0.0471
3	1	0.1111	0.0400
	3	0.1370	0.0309
	5	0.1721	0.0599
5	1	0.1383	0.0593
	3	0.1709	0.0238
	5	0.1371	0.0435

À primeira vista, verifica-se que a grande maioria das *accuracies* obtidas com a replicação dos dados foi inferior às conseguidas sem essa replicação, pelo que é possível constatar que o objectivo primordial desta alteração não foi alcançado. Além disso, contrariamente ao obtido nos resultados da secção 4.4.2, quando o *nbHiddenStates* = 1, as médias das *accuracies* não são iguais.

De forma análoga ao realizado anteriormente, para facilitar a visualização dos resultados, foi gerado o gráfico presente na Figura 4.19. Neste gráfico, para além das *accuracies* dos conjuntos de teste, também são exibidas as *accuracies* dos conjuntos de treino, somente para a posterior análise de *overfitting*. Mais uma vez, as barras verticais, amarelas e a tracejado, delimitam os grupos com o parâmetro *nbHiddenStates* igual.

Pela análise da Figura 4.19, verifica-se que, apesar dos resultados distintos obtidos pelo conjunto de teste quando o *nbHiddenStates* = 1, as *accuracies* dos conjuntos de treino e os seus desvios padrão são iguais, independentemente do valor do parâmetro *windowSize*. Além disso, as *accuracies* dos conjuntos de treino são inferiores às alcançadas pelos conjuntos de teste, o que ainda não se tinha observado até agora. Estes acontecimentos serão analisados posteriormente com maior detalhe. É de notar ainda que, os desvios padrão apresentados pelos conjuntos de treino não se encontram visíveis no gráfico, pois assumem um valor de apenas 0.00014529.

Quando o *nbHiddenStates* = 2, existe um aumento da *accuracy* do conjunto de teste, seguido de uma diminuição do seu valor, à medida que se aumenta o parâmetro *windowSize*. Pelo observado nos resultados dos conjuntos de treino, não é possível afirmar que esta diminuição da *accuracy* tenha ocorrido devido ao *overfitting* dos dados. Quando o *nbHiddenStates* = 3, o valor da *accuracy* cresce com o aumento do *windowSize*, de forma quase linear. Por outro lado, quando o *nbHiddenStates* toma o valor 5, a média da *accuracy* sobe quando se altera o parâmetro *windowSize* de 1 para 3, e decresce

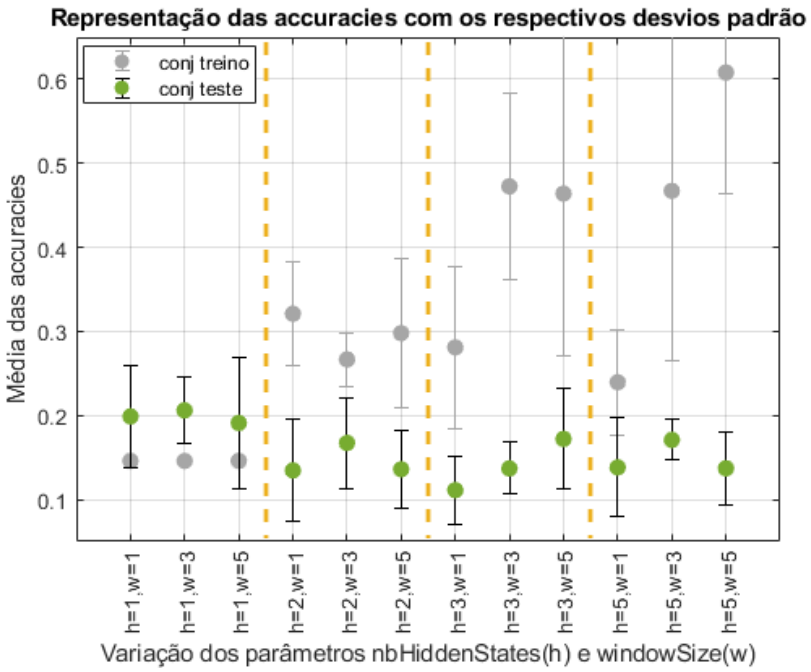


Figura 4.19: Representação gráfica das *accuracies* obtidas com o código HCRF_algorithm_final.m, para o ficheiro [300,300]ms.

quando se aumenta para 5. Neste caso em particular, verifica-se a existência de *overfitting*, uma vez que a *accuracy* do conjunto de treino aumentou e a do conjunto de teste diminui o seu valor, quando se passou do *windowSize* = 3 para *windowSize* = 5. No que diz respeito aos desvios padrão, a maioria dos casos apresenta valores bastante idênticos.

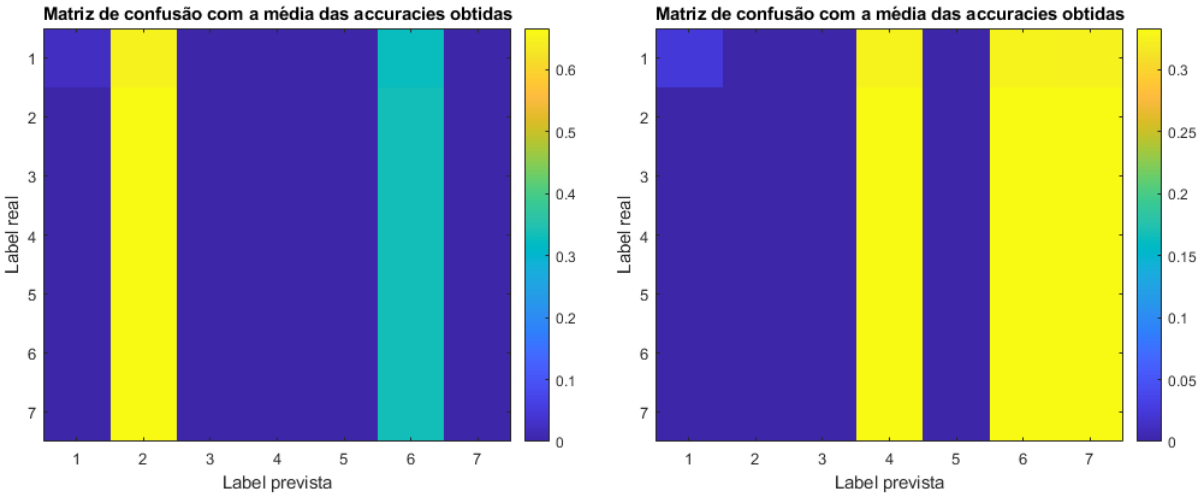
Para consolidar a análise das *accuracies* gerais de cada combinação dos parâmetros *nbHiddenStates* e *windowSize*, é também realizado um estudo às *accuracies* de cada uma das *labels*. Encontra-se, por isso, exposto na Tabela B.7 no Anexo B, todos esses valores das *accuracies*, juntamente com os respectivos desvios padrão. Analisando os resultados obtidos, verifica-se que existem menos zeros nos valores das *accuracies* comparativamente com o atingido no caso sem replicação dos dados (rever Tabela B.5), o que significa que foram classificadas correctamente mais *labels*. Portanto, apesar das *accuracies* gerais não serem superiores às apresentadas nos testes com o algoritmo HCRF sem replicação dos dados, conseguiu-se eliminar a preferência pelas *labels* que continham um maior número de amostras.

Para o *nbHiddenStates* = 1, observa-se que o classificador teve em consideração mais classes na sua previsão do que no caso anterior, onde foram unicamente utilizadas as *labels* 1 e 4. Mais especificamente, quando o *windowSize* = 1, tem-se na primeira e na segunda iteração cerca de 99.6% das amostras classificadas como sendo *label* 2, e na terceira iteração atribuiu-se, à mesma percentagem de amostras, a *label* 4. Posto isto, em cada uma das iterações, acertaram-se em todas as amostras que eram tanto da *label* 2 como da *label* 4. Assim, quando é calculada a média apresentada na Tabela B.7, resultam os valores de *accuracy* de 0.6667 para a *label* 2 e 0.3333 para a *label* 4, aproximadamente, uma vez que em duas das 3 iterações, que correspondem a $2/3$ dos testes, acertaram-se em todas as amostras que eram *label* 2, e nos restantes $1/3$ dos testes acertaram-se em todas as amostras

classificadas com *label* 4. À restante percentagem de amostras, cerca de 0.4%, foi-lhe atribuída a *label* 1. O mesmo ocorreu com o parâmetro *windowSize* = 3. Todavia, em vez de $1/3$ dos testes ter sido classificado com a *label* 4, foi etiquetado com a *label* 6.

Quando *windowSize* = 5, tem-se na primeira iteração 99.6% das amostras classificadas com a *label* 6, na segunda iteração a mesma percentagem de amostras classificadas com a *label* 4, e na última, 99.6% dos dados etiquetados com a *label* 7. Assim, em cada uma das 3 iterações, acertaram-se em todas as amostras das *labels* 4, 6 e 7, o que resulta na média de 0.3333 apresentada na *accuracy* de cada uma destas *labels*.

Apesar disso, com o *nbHiddenStates* = 1, continuam a não ser consideradas todas as *labels* na previsão da classificação dos dados, tal como também se verifica nas matrizes de confusão da Figura 4.20. De forma sucinta, somente as *labels* 2 e 6 são utilizadas na classificação dos dados do caso da Figura 4.20(a) e as *labels* 4, 6 e 7 no caso da Figura 4.20(b). Existe, ainda, uma pequena percentagem de amostras correctamente classificadas com a *label* 1, em ambos os casos.



(a) Matriz de confusão com *windowSize* = 3.

(b) Matriz de confusão com *windowSize* = 5.

Figura 4.20: Matrizes de confusão das *accuracies* do ficheiro [300,300]ms, com o *nbHiddenStates* = 1.

Fixando o parâmetro *windowSize*, verifica-se que, apenas no caso em que este parâmetro toma o valor 3, é que não se obteve nenhum zero nas médias das *accuracies* das *labels*. Embora já se considerem mais *labels* na classificação dos dados, o erro existente nesta classificação ainda é elevado, tal como podemos observar pelos valores dos desvios padrão.

Ainda assim, foi feita uma análise mais pormenorizada dos resultados, de modo a tentar perceber o comportamento de cada *label* em si. Para isso, foram analisadas as matrizes de confusão apresentadas nas Figuras 4.21, 4.22 e 4.23, onde se fixou o parâmetro *nbHiddenStates* com o valor 3 e se variou o valor do *windowSize*.

Na Figura 4.21, verifica-se que o classificador previu quase todos os dados como sendo *label* 5. É agora possível confirmar que as *labels* 3 e 7, que apresentaram zeros na Tabela B.7, foram de facto utilizadas, embora de forma incorrecta. Por exemplo, a *label* 3 classificou dados que na realidade eram *label* 1, e a *label* 7 classificou amostras que na verdade pertenciam às *labels* 3, 4, 5 e 6.

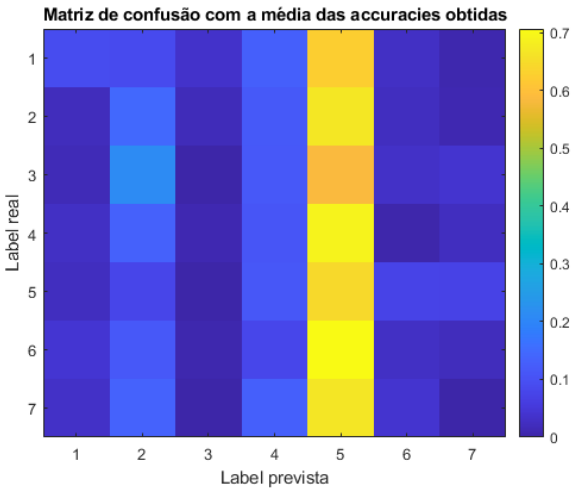


Figura 4.21: Matriz de confusão com as *accuracies* do ficheiro [300,300]ms com $h = 3$ e $w = 1$.

A Figura 4.22 mostra uma melhoria significativa face ao caso anterior, com o *windowSize* = 1. Nesta matriz de confusão, a previsão das classificações foi bastante distribuída, não se destacando nenhuma *label* em particular. É de notar que, neste caso, todas as *labels* foram utilizadas correctamente, em algum momento da previsão. Ainda assim, continua a não existir um destaque da diagonal principal, pois os dados são classificados mais vezes com *labels* incorrectas do que com a correcta. Isto reflecte-se nos valores elevados apresentados pelos desvios padrão.

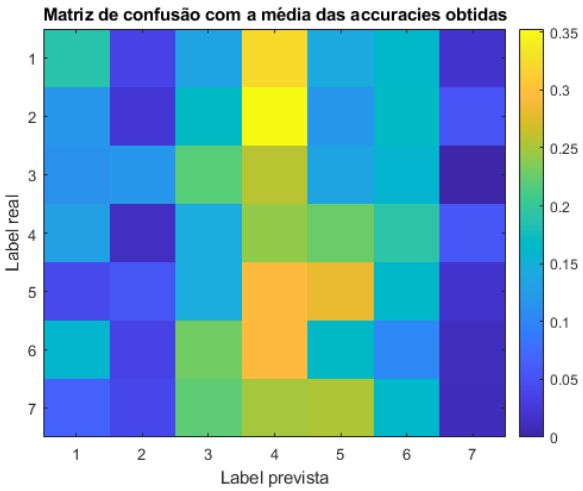


Figura 4.22: Matriz de confusão com as *accuracies* do ficheiro [300,300]ms com $h = 3$ e $w = 3$.

Por último, a Figura 4.23 aparenta ter havido uma regressão na distribuição das *labels* face ao caso anterior. Existe, novamente, uma preferência clara para a *label* 4, seguida das *labels* 2 e 6. Mesmo assim, a *accuracy* geral deste teste foi superior à anterior, com o *windowSize* = 3. Neste caso, também não houve nenhum zero na Tabela B.7, o que significa que todas as *labels* foram utilizadas correctamente, em alguma ocasião.

Posto isto, e no sentido de completar este estudo, segue-se a análise do algoritmo HCRF com replicação dos dados de treino, aplicado ao ficheiro [500,300]ms.

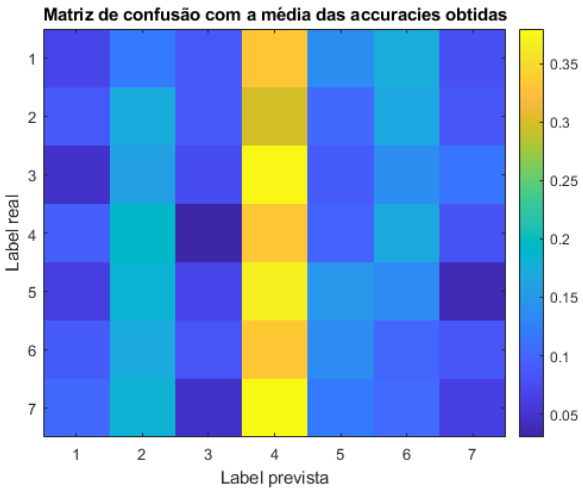


Figura 4.23: Matriz de confusão com as *accuracies* do ficheiro [300,300]ms com $h = 3$ e $w = 5$.

Ficheiro [500,300]ms

Seguindo a mesma linha de raciocínio adoptada nos estudos anteriores, estes novos testes são realizados para o mesmo leque de valores dos parâmetros *windowSize* e *nbHiddenStates*. Posto isto, os resultados das *accuracies* gerais e respectivos desvios padrão, obtidos com o ficheiro [500,300]ms, encontram-se disponibilizados na Tabela 4.6.

Tabela 4.6: Resultados da *accuracy* do conjunto de teste, com o código HCRF_algorithm_final.m, para o ficheiro [500,300]ms.

Parâmetros		**Resultados - Ficheiro [500,300]ms**	
nbHiddenStates	***windowSize***	**Média da *accuracy***	**Desvio padrão da *accuracy***
1	1	0.1827	0.0377
	3	0.0973	0.0688
	5	0.1795	0.1147
2	1	0.1966	0.0479
	3	0.1372	0.0714
	5	0.1850	0.0581
3	1	0.1299	0.0453
	3	0.1407	0.0662
	5	0.1497	0.0521
5	1	0.1287	0.0434
	3	0.1218	0.0119
	5	0.1662	0.0276

Pela análise da Tabela 4.6, é possível observar uma maior oscilação nos valores das médias e dos desvios padrão obtidos, relativamente ao caso anterior sem replicação (ver Tabela 4.4). Esta oscilação abrange valores entre os 0.0973 e os 0.1966 para a média e uma amplitude de 0.1028 nos valores do desvio padrão. Verifica-se também que, de um modo geral, as médias das *accuracies* obtidas diminuíram e alguns desvios padrão atingiram valores relativamente altos, quando comparados com as

respectivas médias.

De forma análoga ao já realizado, foi gerado o gráfico da Figura 4.24, que pretende auxiliar a análise dos resultados obtidos. Quando o *nbHiddenStates* = 1, as médias das *accuracies* dos conjuntos de teste apresentam valores distintos, assim como no caso do ficheiro [300,300]ms com replicação dos dados. Contudo, para o conjunto de treino, tanto as médias como os desvios padrão apresentam o mesmo valor, independentemente do *windowSize*.

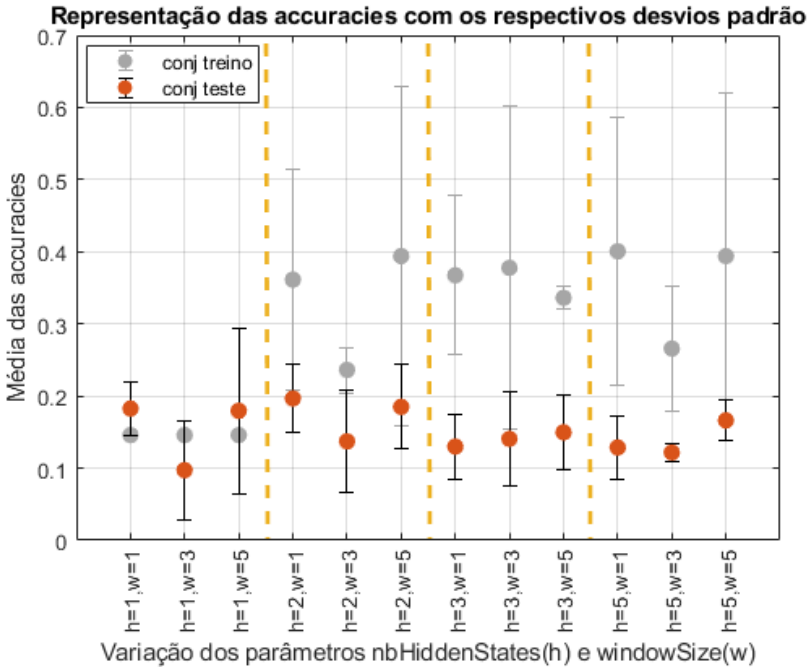


Figura 4.24: Representação gráfica das *accuracies* obtidas com o código HCRF_algorithm_final.m, para o ficheiro [500,300]ms.

À primeira vista, não é possível estabelecer nenhum padrão de variação das *accuracies* com a alteração dos parâmetros *nbHiddenStates* e *windowSize*. Verifica-se que, em nenhum dos casos se obteve uma *accuracy* superior a 0.20, o que contrasta com o obtido para o ficheiro [500,300]ms sem replicação, onde em todos os testes se alcançaram resultados superiores a esse valor. Não obstante, as *accuracies* gerais mais elevadas para este ficheiro foram alcançadas com o *nbHiddenStates* = 2.

Pela análise da Figura 4.24, é possível constatar que, quando o *nbHiddenStates* = 3, também se verifica um aumento das *accuracies* gerais com a subida do valor do parâmetro *windowSize*, semelhante ao observado na Figura 4.19. Relativamente aos outros valores, não há mais conclusões a retirar, além de em nenhum dos testes se ter observado a existência de *overfitting*.

Para completar esta análise, são também investigadas as *accuracies* obtidas por cada uma das *labels*. Estas encontram-se na Tabela B.8 do Anexo B, juntamente com os respectivos desvios padrão.

Numa primeira instância, é possível observar que a quantidade de zeros nas *accuracies* das *labels* diminuiu face ao obtido no ficheiro [300,300]ms (rever Tabela B.7). Fora o *nbHiddenStates* = 1, houve somente uma combinação de valores onde resultou um zero na *label* 5. Isto também demonstra uma melhoria relativamente ao estudo sem a replicação dos dados (rever Tabela B.6), em termos da descentralização das *labels* que possuem um maior número de amostras.

Além disso, os testes com o *nbHiddenStates* = 1 revelam resultados semelhantes aos obtidos com o ficheiro [300,300]ms, em que por cada iteração feita num teste, o classificador foca-se unicamente numa *label*, ou seja, classifica cerca de 99.6% das amostras com essa etiqueta. Contudo, não é sempre a mesma *label* escolhida por iteração, o que resulta nos valores de *acurracy* de 0.3333 ou 0.6667, apresentados na Tabela B.8. Os desvios padrão indicados nesta tabela continuam a revelar valores elevados, o que significa que existe um erro ainda elevado na previsão das classificações.

Analisando com maior detalhe os resultados obtidos com o *nbHiddenStates* = 3, tal como foi feito no ficheiro anterior, encontram-se expostas nas Figuras 4.25, 4.26 e 4.27 as matrizes de confusão referentes a cada um dos três valores de *windowSize* estudados.

Na Figura 4.25, é possível observar que houve uma preferência clara pela *label* 5 na classificação das amostras. Na Tabela B.8, verifica-se que esta *label* possui o valor de *accuracy* mais elevado, mas também o mais alto dos desvios padrão, relativamente às restantes *labels*. Ainda assim, todas as etiquetas conseguiram ter amostras correctamente classificadas, sendo também menor os seus valores de desvio padrão. Tal é possível notar na matriz de confusão nas *labels* 4, 6 e 7, mas não é tão perceptível para as restantes etiquetas, devido aos seus valores reduzidos.

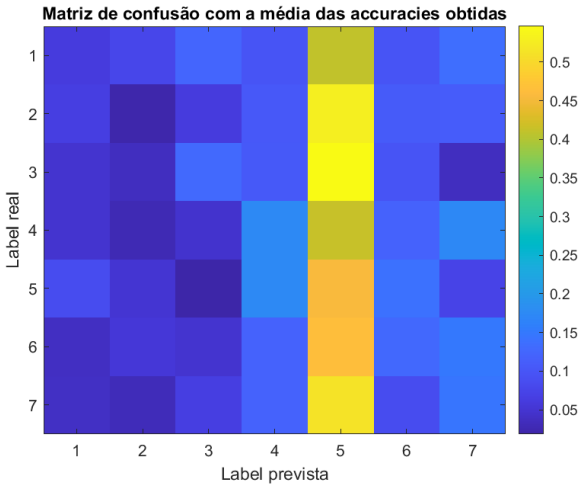


Figura 4.25: Matriz de confusão com as *accuracies* do ficheiro [500,300]ms com $h = 3$ e $w = 1$.

Relativamente à matriz de confusão da Figura 4.26, é possível verificar que o aumento do parâmetro *windowSize* levou o foco do classificador para as *labels* 3 e 7, tendo a maioria das amostras sido classificadas com estas duas etiquetas. Isto também é visível pela Tabela B.8, que mostra valores elevados de *accuracy* e de desvio padrão para ambas as *labels*. De forma análoga ao caso anterior com o *windowSize* = 1, todas as *labels* foram correctamente utilizadas em algum momento da classificação, o que não é muito perceptível pela Figura 4.26, devido aos seus valores reduzidos.

Por último, a Figura 4.27 revela uma melhoria dos resultados comparando aos dois casos anteriores, com diferentes valores de *windowSize*. Verifica-se uma distribuição mais equilibrada das *labels* pelos dados testados. As *accuracies* das *labels* 2, 3, 4 e 5 subiram face ao caso com o *windowSize* = 3, assim como os seus desvios padrão. Esta matriz de confusão mostra mais semelhanças com a exibida na Figura 4.22, onde também houve uma distribuição maior das *labels* pelos dados.

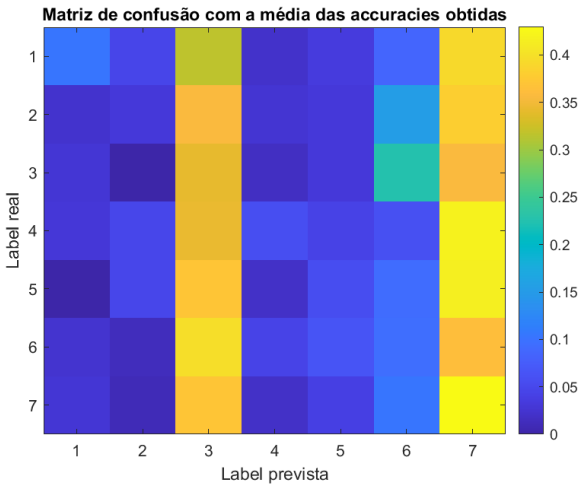


Figura 4.26: Matriz de confusão com as *accuracies* do ficheiro [500,300]ms com $h = 3$ e $w = 3$.

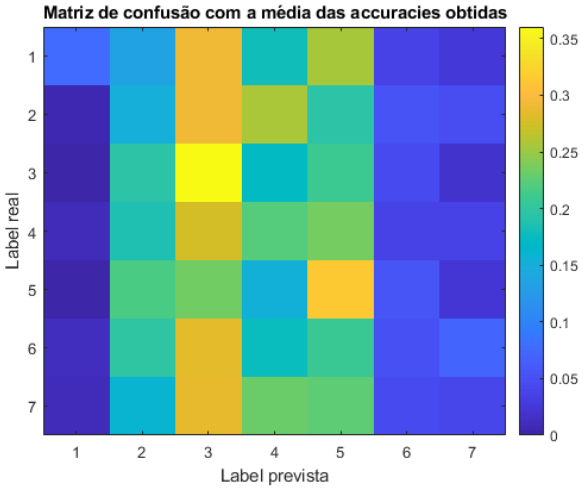


Figura 4.27: Matriz de confusão com as *accuracies* do ficheiro [500,300]ms com $h = 3$ e $w = 5$.

A subida da *accuracy* geral demonstrada nestes 3 testes deve-se ao facto de terem sido utilizadas mais *labels* na previsão das classificações. O ideal seria que houvesse um destaque da diagonal principal, o que não ocorreu em nenhuma das situações ilustradas.

4.5 Comparação e análise de resultados

Em adição às breves comparações efectuadas ao longo da descrição dos resultados, pretende-se, nesta secção, desenvolver comparações mais abrangentes, no sentido de consolidar as conclusões a retirar desta dissertação. Para isso, é feita a análise dos resultados obtidos com os 3 algoritmos estudados. Além disso, investigam-se os resultados peculiares atingidos com o parâmetro *nbHiddenStates* = 1.

4.5.1 Igualdade dos resultados com o parâmetro *nbHiddenStates* = 1

Nos testes desenvolvidos com o algoritmo HCRF sem replicação de dados, observa-se que, apesar da variação do *windowSize* e do tamanho do ficheiro de entrada, os resultados das *accuracies* são sempre iguais, quando o parâmetro *nbHiddenStates* toma o valor 1.

Posto isto, foram executados novos testes com outros ficheiros de entrada, de tamanho diferentes. Na Tabela 4.7 estão indicados os resultados que se obtiveram anteriormente, na secção 4.4.2, juntamente com os novos resultados atingidos pelos ficheiros [750,300]ms, [1000,300]ms e [1000,1000]ms.

Tabela 4.7: Resultados das *accuracies* de vários ficheiros de dados, com o *nbHiddenStates* = 1.

Ficheiros de dados	**Parâmetro *windowSize***	**Resultados**	
		Média da *accuracy*	**Desvio padrão da *accuracy***
[300,300]ms	1	0.2529	0.0240
	3	0.2529	0.0240
	5	0.2529	0.0240
[500,300]ms	1	0.2529	0.0240
	3	0.2529	0.0240
	5	0.2529	0.0240
[750,300]ms	1	0.2529	0.0240
	3	0.2529	0.0240
	5	0.2529	0.0240
[1000,300]ms	1	0.2541	0.0252
	3	0.2541	0.0252
	5	0.2541	0.0252
[1000,1000]ms	1	0.2541	0.0252
	3	0.2541	0.0252
	5	0.2541	0.0252

Portanto, com a análise da Tabela 4.7, verifica-se que, de facto, os valores das *accuracies* variam consoante o tamanho do ficheiro de dados. Ainda assim, essa variação é muito reduzida. É importante notar que os resultados se encontram arredondados a 4 casas decimais mas, eliminando esse arredondamento, obtêm-se exactamente os mesmos valores.

Tal como já tinha sido referido anteriormente (ver secção 4.4.2), no ficheiro [300,300]ms cerca de 99.65% das observações foram classificadas com a *label* 4, sendo que as restantes foram previstas com a *label* 1. Esta percentagem aplica-se também aos ficheiros [500,300]ms e [750,300]ms, o que se traduz numa *accuracy* geral igual para estes 3 ficheiros.

Por outro lado, verifica-se que, apesar da *accuracy* do ficheiro [1000,300]ms ser igual à obtida com o ficheiro [1000,1000]ms, as previsões das classificações foram ligeiramente diferentes. Neste último, foram classificadas com a *label* 4 cerca de 99.06% das observações e as restantes com a *label* 1. Isto demonstra uma subtil redução face ao obtido com o ficheiro [1000,300]ms, que revelou uma percentagem de 99.53% de observações classificadas com a *label* 4. Embora exista esta ligeira diferença, classificaram-se correctamente o mesmo número de amostras, pelo que as *accuracies* gerais, e res-

pectivos desvios padrão, apresentam o mesmo valor.

Relativamente à variação do *windowSize*, foram também executados novos testes com o ficheiro [500,300]ms e com outros valores para este parâmetro. Os novos resultados obtidos, com o *windowSize* a tomar os valores 20 e 50, revelam que as *accuracies* são exactamente iguais às obtidas com os outros 3 valores analisados anteriormente. Mais especificamente, foram classificadas de forma correcta o mesmo número de observações, com as *labels* 1 e 4. Contudo, não se considera legítimo concluir que os resultados das *accuracies* são sempre iguais mesmo com a variação do parâmetro *windowSize*, pois podem existir outros factores a influenciar estes resultados, como por exemplo, o *overfitting* e os próprios dados testados.

No entanto, no estudo do algoritmo HCRF com replicação dos dados de treino, obtiveram-se valores de *accuracy* diferentes para os conjuntos de testes, mas exactamente iguais nos conjuntos de treino. Partindo da explicação mais pormenorizada dada na secção 4.4.4, verifica-se que, em cada iteração do método *k-fold*, havia sempre uma *label* que se destacava, classificando cerca de 99.6% do total de amostras. Como o número de dados de cada *label* variava consoante o conjunto de teste em questão, também a *label* "em destaque" alterava, o que resultou na diferença dos valores das *accuracies* gerais observada.

Deste modo, é possível constatar que, quando o parâmetro *nbHiddenStates* = 1, existe um foco elevado em apenas uma das classes e, embora as *accuracies* obtidas com este parâmetro sejam, na maioria das vezes, superiores aos restantes resultados obtidos com outros valores de *nbHiddenStates*, estes não serão considerados nas comparações finais. Os modelos obtidos cingem-se demasiado a uma *label* isolada, ignorando as restantes. Tal não é o pretendido neste trabalho que tenciona abranger várias *labels*, em simultâneo, na classificação dos seus dados.

4.5.2 Algoritmos kNN, CRF e HCRF

Observando de forma superficial os resultados obtidos com os algoritmos kNN, CRF e HCRF, apresentados nas Tabelas 4.1, 4.2, 4.3 e 4.4, é possível verificar que cada um destes algoritmos demonstra desempenhos diferentes nas previsões das classificações dos seus dados.

Para o mesmo conjunto de ficheiros de dados, o algoritmo kNN revelou *accuracies* entre os 0.17 e os 0.19, nunca atingindo valores acima deste limite máximo, enquanto que o algoritmo CRF mostrou os resultados das *accuracies* entre os 0.18 e os 0.22, com qualquer um dos valores de *windowSize* testados. Em duas ocasiões, observaram-se ainda valores acima dos 0.22, com os valores de *windowSize* de 1 e 3. Posto isto, pode-se afirmar que o algoritmo CRF mostrou um melhor desempenho nas suas previsões do que o algoritmo kNN.

Focando somente o algoritmo CRF, é possível observar que, quando o *windowSize* toma o valor 1, obtém-se 9 vezes valores de *accuracy* acima dos 0.21. Por outro lado, quando este parâmetro toma o valor 0, encontra-se apenas 1 resultado com a *accuracy* superior ao limite em análise e 2 resultados quando o *windowSize* toma o valor 3. Além disso, e recuando até à secção 4.3.2, é possível constatar que esta diferença nas *accuracies*, obtidas com os valores de *windowSize* 1 e 3, é possivel-

mente causada pela existência de *overfitting*. Deste modo, pode-se afirmar que o algoritmo CRF com o *windowSize* = 1 obteve melhores resultados, na sua generalidade, do que os restantes dois valores testados para este parâmetro.

Na aplicação do algoritmo HCRF foram utilizados dois ficheiros de dados, referentes aos intervalos [300,300]ms e [500,300]ms. Nenhum destes dois ficheiros apresentou os melhores ou os piores resultados atingidos nos algoritmos kNN e CRF. Para auxiliar esta comparação, sintetizaram-se os resultados obtidos nas Tabelas 4.8 e 4.9. A média da *accuracy* indicada no algoritmo CRF corresponde à mais elevada atingida entre os 3 valores de *windowSize* testados. Relativamente ao HCRF, é exibida a média da *accuracy* mais alta e a mais baixa, com a respectiva combinação dos parâmetros estudados. Relembra-se ainda que o parâmetro *nbHiddenStates* está representado por h e o *windowSize* por w.

A partir da Tabela 4.8, verifica-se que o pior valor obtido com o HCRF é bastante próximo do melhor alcançado pelo CRF. Quando se passa para o resultado mais elevado atingido pelo HCRF, é possível verificar que se distancia dos restantes e que, em nenhum momento do estudo com o algoritmo CRF, com os diferentes ficheiros, se conseguiu a *accuracy* de 0.24.

Tabela 4.8: Resultados obtidos com os 3 algoritmos para o ficheiro [300,300]ms.

Algoritmos de classificação	**Parâmetros**	**Média da *accuracy***
kNN	-	0.1866
CRF	w = 3	0.2083
HCRF	h = 5 w = 5	0.2059
	h = 3 w = 3	0.2494

Relativamente ao obtido com o ficheiro [500,300]ms exibido na Tabela 4.9, observa-se que os resultados são bastante idênticos aos revelados na Tabela 4.8. O pior resultado do algoritmo HCRF é próximo do conseguido pelo CRF e o melhor também se encontra na casa dos 0.24, distanciando-se dos restantes.

Tabela 4.9: Resultados obtidos com os 3 algoritmos para o ficheiro [500,300]ms.

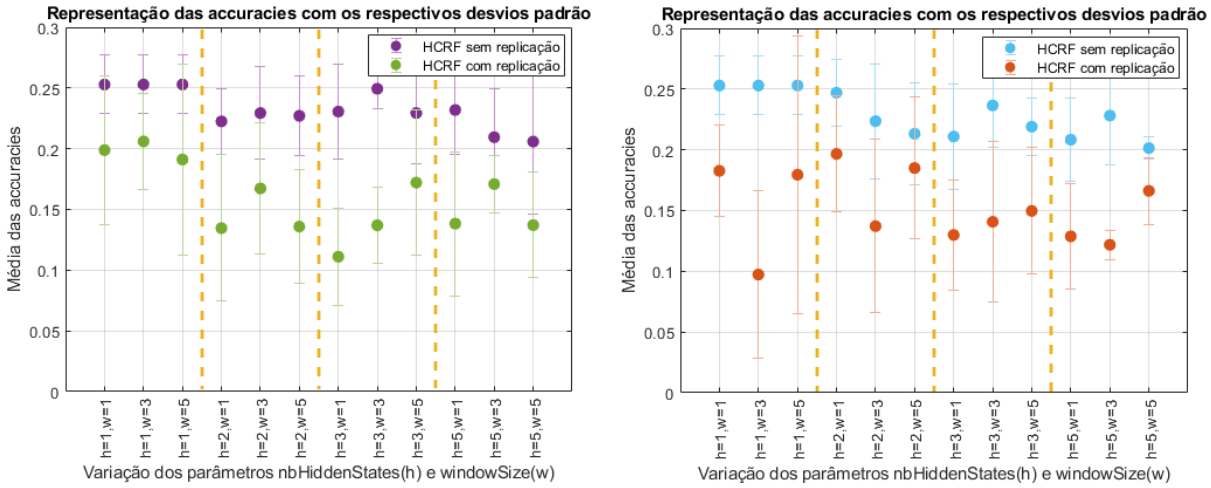
Algoritmos de classificação	**Parâmetros**	**Média da *accuracy***
kNN	-	0.1835
CRF	w = 3	0.2059
HCRF	h = 5 w = 5	0.2014
	h = 2 w = 1	0.2470

Além disso, é importante notar que, apesar da proximidade dos valores dos melhores resultados obtidos, em cada um dos ficheiros, estes foram alcançados com valores de *nbHiddenStates* e de *windowSize* distintos. Isto transmite a ideia de que para cada ficheiro com tamanho diferentes, é necessário encontrar a melhor combinação de valores destes dois parâmetros. Adicionalmente, e como seria de

esperar, em ambos os casos, o algoritmo kNN apresentou o pior desempenho dos 3 algoritmos de classificação estudados.

4.5.3 Algoritmo HCRF com e sem replicação de dados

Nas Figuras 4.28(a) e 4.28(b) estão exibidas as *accuracies* alcançadas pelos ficheiros [300,300]ms e [500,300]ms, respectivamente. Estes valores dizem respeito aos resultados obtidos com os conjuntos de teste, com e sem replicação dos dados utilizados no treino do classificador (ver legendas das figuras).



(a) Resultados das *accuracies* do ficheiro (300,300)ms.

(b) Resultados das *accuracies* do ficheiro (500,300)ms.

Figura 4.28: Representação gráfica das *accuracies* obtidas com e sem replicação dos dados.

À primeira vista, observa-se que, em ambos os casos, as *accuracies* mais elevadas foram sempre obtidas sem a replicação dos dados. Ainda assim, em algumas situações, existiu a sobreposição dos intervalos dos desvios padrão. Os resultados obtidos sem a replicação mostram uma menor oscilação dos seus valores, além de revelarem valores de desvio padrão inferiores. Comparando agora o tamanho dos ficheiros de dados, não é perceptível a vantagem entre qualquer um dos ficheiros. Ambos têm as suas *accuracies* entre os 0.20 e 0.25, e desvios padrão com intervalos semelhantes. É ainda de notar que, no caso do ficheiro [500,300]ms, as *accuracies* mais elevadas para os casos com e sem replicação foram atingidas com a mesma combinação de parâmetros *nbHiddenStates* e *windowSize*. O mesmo não se verificou para o ficheiro [300,300]ms.

O objectivo principal da replicação dos dados nos conjuntos de treino, de aumentar as *accuracies* gerais dos testes, não foi atingido. Contudo, pelo que se averiguou nas análises realizadas ao longo deste capítulo, conseguiu-se ter em consideração mais classes durante o processo de previsão. Quer isto dizer que o equilíbrio no número de amostras por *label*, permitiu ao classificador não se focar em demasia em *labels* específicas, ignorando as restantes. Devido a este facto, a replicação mostrou então uma vantagem face à aplicação do algoritmo HCRF sem qualquer manipulação.

É ainda importante salientar que o estudo com a replicação dos dados no conjunto de treino não foi efectuado para o algoritmo CRF, uma vez que os resultados são efectivamente melhores com o

algoritmo HCRF e a replicação deverá provocar o mesmo padrão de resultados no algoritmo CRF.

4.5.4 Outras abordagens

Outras estratégias poderiam ter sido testadas, no sentido de tentar obter melhores valores de *accuracy*. Além da replicação do conjunto de treino, podia ter sido feita a eliminação da(s) *label(s)* com o(s) menor(es) número(s) de amostras, reduzir o número de amostras dos conjuntos de treino até ao seu valor mínimo, ou até efectuar a replicação mas com uma pequena quantidade de ruído, de modo a não ser uma repetição integral de certas amostras dos dados de treino.

Posto isto, foi seleccionada a melhor combinação de parâmetros para cada um dos ficheiros, indicada anteriormente nas Tabelas 4.8 e 4.9, e foram testados estes 3 casos adicionais: HCRF com replicação e com ruído, HCRF com redução do conjunto de treino e HCRF com eliminação de *labels*.

Portanto, para a adição de ruído, foi primeiramente calculada uma distribuição normal dos dados de cada *feature*, para cada uma das amostras do conjunto de treino. A partir da média e do desvio padrão da distribuição normal encontrada, é gerado um *normal random number*, através da função *normrnd* no MATLAB. Ao valor da média da distribuição é subtraído este *normal random number*, criando assim um factor aleatório que corresponderá ao ruído. Este valor é, posteriormente, somado a cada um dos dados das *features*. Desta forma, o ruído simula a aquisição de uma nova amostra, semelhante às anteriores.

No que diz respeito à redução do conjunto de treino, o processo é semelhante ao realizado na replicação. Todavia, em vez de se encontrar a *label* com o maior número de amostras no conjunto de treino, procura-se a *label* que possui a menor quantidade. Este valor é definido como o número de amostras por *label* e, por isso, eliminam-se as restantes amostras do conjunto de treino quando se ultrapassa este valor por *label*. Isto permite que haja um equilíbrio no número de observações por *label*, sem haver a necessidade de repetir dados.

Relativamente à eliminação de *labels*, tal como é possível observar pela Tabela 3.3, tanto a *label* 3 como a 5 possuem um número de amostras bastante inferior às restantes. Deste modo, foram eliminadas todas as amostras com estas duas *labels*, passando a existir apenas 5 *labels* para a classificação dos dados.

Os resultados obtidos para os novos testes estão apresentados nas Tabelas 4.10 e 4.11. Para o ficheiro [300,300]ms foram utilizados os parâmetros *nbHiddenStates* e *windowSize* com o valor 3, e no caso do ficheiro [500,300]ms, com o *nbHiddenStates* = 2 e o *windowSize* = 1.

Tal como é possível observar, em ambos os casos, os melhores resultados foram obtidos com a eliminação dos dados das *labels* com o menor número de amostras, sendo que, para esta situação, não houve qualquer manipulação dos dados. A redução dos dados dos conjuntos de treino não se revelou tão vantajosa, pois o conjunto de dados pode ter ficado bastante reduzido, levando a que o treino do classificador não fosse muito eficaz. Como a diferença entre o número de amostras era significativa, muito provavelmente os conjuntos de treino ficaram com uma dimensão muito reduzida. Seria, por isso, necessário ter mais dados para utilizar no treino. Por sua vez, o ruído revelou uma

Tabela 4.10: Resultados do ficheiro [300,300]ms, para diferentes abordagens com o algoritmo HCRF.

Estratégia	**Alterações**	**Média da *accuracy***	**Desvio padrão da *accuracy***
HCRF (sem manipulação)	-	0.2494	0.0167
HCRF com replicação (com e sem ruído)	Conjunto de treino	0.1370	0.0309
	Conjunto de treino com adição de ruído	0.1954	0.0436
HCRF com redução	Conjunto de treino	0.1555	0.0978
HCRF com eliminação de *labels*	Totalidade dos dados	0.2754	0.0331

Tabela 4.11: Resultados do ficheiro [500,300]ms, para diferentes abordagens com o algoritmo HCRF.

Estratégia	**Alterações**	**Média da *accuracy***	**Desvio padrão da *accuracy***
HCRF (sem manipulação)	-	0.2470	0.0273
HCRF com replicação (com e sem ruído)	Conjunto de treino	0.1966	0.0479
	Conjunto de treino com adição de ruído	0.2588	0.0137
HCRF com redução	Conjunto de treino	0.2094	0.0923
HCRF com eliminação de *labels*	Totalidade dos dados	0.2938	0.0114

melhoria na *accuracy* face ao caso com a replicação integral dos dados. Isto mostra que a replicação de dados com valores de *features* próximas das reais foi vantajoso para a previsão das classificações. Deste modo, caso se pretendesse manter as 7 *labels*, esta seria a abordagem mais indicada.

Além disso, e como se constatou anteriormente, cada conjunto de dados tem a sua combinação de parâmetros, ou seja, apesar de estes testes terem sido realizados para a melhor combinação de parâmetros do HCRF sem manipulação, estes podem não ser a melhor combinação para as outras abordagens. Tal é visível directamente no ficheiro [300,300]ms com o HCRF com replicação, que não obteve os seus melhores resultados com a combinação de parâmetros apresentada, *nbHiddenStates* = 3 e *windowSize* = 3.

Comparando novamente o tamanho dos ficheiros, já é possível verificar que, de um modo geral, o ficheiro que abrange um maior intervalo de tempo atinge melhores valores de *accuracy* do que o outro ficheiro estudado, com um intervalo menor.

Capítulo 5

Conclusão

Com esta dissertação, pretendia-se gerar um modelo de um classificador que conseguisse prever quais os *feedbacks* motivacionais a transmitir aos idosos, em determinados momentos de um jogo. Os dados foram obtidos a partir de sessões integradas pelos idosos a jogarem os *exergames* e os fisioterapeutas a fornecerem as indicações necessárias. Os *feedbacks* motivacionais seleccionados para a replicação correspondem àqueles com um maior número de amostras, de modo a contribuir para o treino dos classificadores.

O estudo baseou-se em 3 algoritmos de classificação distintos: kNN, CRF e HCRF. Depois de uma análise pormenorizada dos resultados obtidos com cada um dos algoritmos, conclui-se que aquele que apresenta um melhor desempenho é o HCRF. Posteriormente, procurou-se atingir melhores valores de *accuracy*, o que conduziu à realização de alterações na implementação do algoritmo HCRF. Estas alterações consistiam em replicar os dados utilizados no treino do classificador, tendo sido realizado um estudo detalhado desta situação. Embora não se tenham verificado melhorias na *accuracy* dos resultados, esta replicação mostrou que, com o equilíbrio do número de amostras por *label*, existe uma distribuição mais uniforme das *labels* que são usadas na previsão, não dando preferências àquelas que possuem um maior número de amostras.

Verificou-se também que cada ficheiro de dados revela resultados distintos, consoante o conjunto de parâmetros *nbHiddenStates* e *windowSize* seleccionado. Além disso, a mesma combinação destes parâmetros pode não ser a ideal para o mesmo ficheiro, com conjuntos de treino e de teste diferentes. É, por isso, necessário testar vários valores nestes parâmetros, de modo a tentar alcançar uma combinação que produza bons resultados. Todavia, ao longo deste estudo, é possível notar que, por vezes, a existência de *overfitting* ocorria quando o parâmetro *windowSize* tomava o valor 5.

Outras abordagens poderiam ter sido adoptadas no sentido de melhorar as *accuracies* obtidas. Algumas foram identificadas, mas não foram minuciosamente estudadas. Entre elas destaca-se a eliminação das *labels* com um número de amostras baixo relativamente às restantes, que alcançou os valores de *accuracy* mais elevados neste trabalho. Os resultados obtidos, quando se eliminaram as *labels* 3 e 5, confirmaram um dos grandes desafios enfrentados, pois a discrepância entre o número de amostras por *label*, levou a que, na grande maioria das vezes, houvesse uma preferência para aquelas

que possuíam um maior número de amostras, sendo as restantes desprezadas pelo classificador. Contudo, e mantendo as 7 *labels*, conclui-se que a melhor opção seria realizar a replicação dos dados de treino com uma pequena porção de ruído. Assim, os dados replicados são próximos mas não exactamente iguais aos originais, o que revelou ser uma vantagem, pois os resultados das *accuracies* foram mais elevados para este caso, comparando com os outros que também utilizavam as 7 *labels*.

Embora, de um modo geral, uma *accuracy* de 0.2588 possa parecer baixa, é mais elevada do que fazendo a previsão das *labels* de forma aleatória. Mais especificamente, se a previsão das 7 *labels* fosse aleatória, a probabilidade de sucesso seria, aproximadamente, 0.14 ($1/7 \quad 0.1428571$), pelo que este classificador já se mostra vantajoso.

5.1 Trabalho futuro

Uma das dificuldades encontradas durante a realização desta dissertação consistiu na etiquetagem das transcrições dos jogos. A divisão feita pelos 3 tipos de *feedback* (correctivo, explicativo e motivacional) deveria ter sido realizada por mais do que uma pessoa, devido ao seu carácter subjectivo. Assim, pode ter-se tornado tendenciosa, uma vez que havia mensagens que encaixavam em mais do que um tipo de *feedback*. As diferenças entre as mensagens que permitiam distinguir qual o seu propósito eram, por vezes, ténues, o que resultou numa dificuldade acrescida.

Outras decisões poderiam ter sido tomadas aquando da selecção dos dados de estudo. Por exemplo, a escolha dos parâmetros provenientes dos jogos e das articulações lidas pela *Kinect* podia ter sido diferente da seleccionada. Adicionalmente, também se pode testar mais ou menos parâmetros e analisar a influência que isso provocaria nos resultados.

Além disso, podem ser realizados mais testes com o algoritmo HCRF para ficheiros de dados com outras dimensões, algo que não fez parte das experiências efectuadas neste estudo. O conjunto dos parâmetros utilizados no algoritmo HCRF (*windowSize* e *nbHiddenStates*) também pode ser estendido, principalmente se se aumentar o tamanho do ficheiro de dados. O *windowSize* não deverá ser necessário aumentar muito mais, pois em alguns casos já se observou a existência de *overfitting*. Contudo, o mesmo não se aplica para o *nbHiddenStates*.

Num trabalho futuro, sugere-se também que se aprofunde a análise das últimas abordagens indicadas na secção 4.5.4, uma vez que algumas pareceram promissoras, em termos da melhoria da *accuracy* do classificador.

Referências

[1] J. A. G. Marin, K. F. Navarro, and E. Lawrence. Serious games to improve the physical health of the elderly: A categorization scheme. In *CENTRIC 2011, The Fourth International Conference on Advances in Human-Oriented and Personalized Mechanisms, Technologies, and Services*, pages 64–71. IARIA XPS Press, Outubro 2011.

[2] U. N. DESA. World population prospects 2019: Highlights. *New York (US): United Nations - Department for Economic and Social Affairs, Population Division*, 2019.

[3] L. H. Larsen, L. Schou, H. H. Lund, and H. Langberg. The physical effect of exergames in healthy elderly - a systematic review. *Games for Health Journal*, 2(4):205–212, 2013.

[4] C. I. Johnson, S. K. T. Bailey, and W. L. V. Buskirk. *Designing Effective Feedback Messages in Serious Games and Simulations: A Research Review*, pages 119–140. Springer, Cham, 2017.

[5] T. Baranowski, K. Bower, P. Krebs, C. Lamoth, and E. J. Lyons. Effective feedback procedures in games for health. *Games For Health Journal*, 2(6):320–326, Dezembro 2013.

[6] C. Lamoth, R. Alingh, and S. R. Caljouw. Exergaming for elderly: Effects of different types of game feedback on performance of a balance task. *Studies in Health Technology and Informatics*, 181: 103–107, Setembro 2012.

[7] D. Drummond, A. Hadchouel, and A. Tesnière. Serious games for health: three steps forwards. *Advances in Simulation*, 2(3):1–9, Dezembro 2017.

[8] C. Burgers, A. Eden, M. D. van Engelenburg, and S. Buningh. How feedback boosts motivation and play in a brain-training game. *Computers in Human Behavior*, 48:94–103, Julho 2015.

[9] N. Zeng, Z. Pope, J. E. Lee, and Z. Gao. A systematic review of active video games on rehabilitative outcomes among older patients. *Journal of Sport and Health Science*, 6(1):33–43, 2017.

[10] C. M. Bleakley, D. Charles, A. Porter-Armstrong, M. D. McNeill, S. M. McDonough, and B. McCormack. Gaming for health: A systematic review of the physical and cognitive effects of interactive computer games in older adults. *Journal of Applied Gerontology*, 34(3):166–189, Abril 2015.

[11] D. Djaouti, J. Alvarez, J. P. Jessel, and O. Rampnoux. *Origins of Serious Games*, pages 25–43. Springer, Londres, 2011.

[12] F. Laamarti, M. Eid, and A. E. Saddik. An overview of serious games. *International Journal of Computer Games Technology*, 2014:1–15, Outubro 2014. Article ID 358152.

[13] P. Rego, P. M. Moreira, and L. P. Reis. Serious games for rehabilitation: A survey and a classification towards a taxonomy. In *5th Iberian Conference on Information Systems and Technologies*, pages 1–6. IEEE, 2010.

[14] Games for Health Europe. `https://www.gamesforhealtheurope.org/`. Acedido a 2020-06-28.

[15] N. Shin, L. M. Sutherland, C. A. Norris, and E. Soloway. Effects of game technology on elementary student learning in mathematics. *British Journal of Educational Technology*, 43(4):540–560, Julho 2012.

[16] W. R. Watson, C. J. Mong, and C. A. Harris. A case study of the in-class use of a video game for teaching high school history. *Computers Education*, 56(2):466–474, Fevereiro 2011.

[17] J. J. Shepherd, R. J. Doe, M. Arnold, N. Cheek, Y. Zhu, and J. Tang. Lost in the middle kingdom: A second language acquisition video game. In *Proceedings of the 49th Annual Southeast Regional Conference*, ACM-SE '11, page 290–294. Association for Computing Machinery, Março 2011.

[18] S. Arnab, P. Petridis, I. Dunwell, and S. de Freitas. *Enhancing Learning in Distributed Virtual Worlds through Touch: A Browser-based Architecture for Haptic Interaction*, pages 149–167. Springer, Londres, 2011.

[19] J. J. Lin, L. Mamykina, S. Lindtner, G. Delajoux, and H. B. Strub. Fish'n'steps: Encouraging physical activity with an interactive computer game. In P. Dourish and A. Friday, editors, *Proceedings of the 8th international conference on Ubiquitous Computing*, UbiComp '06, pages 261–278, Berlin, Heidelberg, Setembro 2006. Springer-Verlag.

[20] A. Whitehead, H. Johnston, N. Nixon, and J. Welch. Exergame effectiveness: What the numbers can tell us. In *Proceedings of the 5th ACM SIGGRAPH Symposium on Video Games*, Sandbox '10, page 55–62. Association for Computing Machinery, Julho 2010.

[21] J. A. McKanna, H. Jimison, and M. Pavel. Divided attention in computer game play: Analysis utilizing unobtrusive health monitoring. In *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 6247–6250. IEEE, Setembro 2009.

[22] M. S. Cameirão, S. B. i Badia, E. D. Oller, and P. F. Verschure. The rehabilitation gaming system: a review. *Studies in Health Technology and Informatics*, 145(6):65–83, Fevereiro 2009.

[23] E. Flores, G. Tobon, E. Cavallaro, F. I. Cavallaro, J. C. Perry, and T. Keller. Improving patient motivation in game development for motor deficit rehabilitation. In *Proceedings of the 2008 International Conference on Advances in Computer Entertainment Technology*, ACE '08, page 381–384. Association for Computing Machinery, Dezembro 2008.

[24] S. T. Smith, A. Talaei-Khoei, M. Ray, and P. Ray. Electronic games for aged care and rehabilitation. In *2009 11th International Conference on e-Health Networking, Applications and Services (Healthcom)*, pages 42–47. IEEE, Dezembro 2009.

[25] J. W. Burke, M. McNeill, D. K. Charles, P. J. Morrow, J. H. Crosbie, and S. M. McDonough. Optimising engagement for stroke rehabilitation using serious games. *The Visual Computer*, 25(12): 1085–1099, Dezembro 2009.

[26] K. Tsiakas, M. Kyrarini, V. Karkaletsis, F. Makedon, and O. Korn. A taxonomy in robot-assisted training: Current trends, needs and challenges. *Technologies*, 6(4):1–19, 2018.

[27] P. Kan, R. Huq, J. Hoey, R. Goetschalckx, and A. Mihailidis. The development of an adaptive upper-limb stroke rehabilitation robotic system. *Journal of NeuroEngineering and Rehabilitation*, 8(33): 1–18, 2011.

[28] G. Gordon, S. Spaulding, J. K. Westlund, J. J. Lee, L. Plummer, M. Martinez, M. Das, and C. L. Breazeal. Affective personalization of a social robot tutor for children's second language skills. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI '16, pages 3951–3957. AAAI Press, Fevereiro 2016.

[29] J. K. Westlund, J. J. Lee, L. Plummer, F. Faridi, J. Gray, M. Berlin, H. Quintus-Bosz, R. Hartmann, M. Hess, S. Dyer, K. dos Santos, S. Örn Aalgeirsson, G. Gordon, S. Spaulding, M. Martinez, M. Das, M. Archie, S. Jeong, and C. Breazeal. Tega: a social robot. In *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 561–561. IEEE, Março 2016.

[30] K. Andrade, G. Fernandes, G. A. Caurin, A. A. Siqueira, R. A. F. Romero, and R. Pereira. Dynamic player modelling in serious games applied to rehabilitation robotics. In *Proceedings of the 2014 Joint Conference on Robotics: SBR-LARS Robotics Symposium and Robocontrol*, pages 211–216. IEEE, Outubro 2014.

[31] D. Siewiorek, A. Smailagic, and A. Dey. Architecture and applications of virtual coaches. *Proceedings of the IEEE*, 100(8):2472–2488, 2012.

[32] F. Ofli, G. Kurillo, Štěpán Obdržálek, R. Bajcsy, H. B. Jimison, and M. Pavel. Design and evaluation of an interactive exercise coaching system for older adults: Lessons learned. *IEEE Journal of Biomedical and Health Informatics*, 20(1):201–212, 2016.

[33] S. Scott. *ABLE Bodies Balance Training*. Human Kinetics, 2008.

[34] M. Duff, Y. Chen, L. Cheng, S. M. Liu, P. Blake, S. L. Wolf, and T. Rikakis. Adaptive mixed reality rehabilitation improves quality of reaching movements more than traditional reaching therapy following stroke. *Neurorehabilitation and Neural Repair*, 27(4):306–315, 2013.

[35] Augmented Human Assistant. `http://aha.isr.tecnico.ulisboa.pt/`, . Acedido a 2020-08-20.

[36] M. Čaić, J. Avelino, D. Mahr, G. Odekerken-Schröder, and A. Bernardino. Robotic versus human coaches for active aging: An automated social presence perspective. *International Journal of Social Robotics*, 12(4):867–882, 2020.

[37] J. Muñoz, A. Gonçalves, M. Cameirão, S. B. i Badia, and E. Gouveia. Measured and perceived physical responses in multidimensional fitness training through exergames in older adults. In *2018 10th International Conference on Virtual Worlds and Games for Serious Applications (VS-Games)*, pages 1–4. IEEE Computer Society, Setembro 2018.

[38] A. Gonçalves, J. Muñoz, E. Gouveia, M. Cameirão, and S. B. i Badia. Portuguese tradition inspired exergames for older people - strategic tools to promote functional fitness. In *Proceedings of the 5th International Congress on Sport Sciences Research and Technology Support (icSPORTS 2017)*. SCITEPRESS – Science and Technology Publications, Novembro 2017.

[39] F. Ahmed, P. P. Paul, and M. L. Gavrilova. Dtw-based kernel and rank-level fusion for 3d gait recognition using kinect. *The Visual Computer*, 31(6-8):915–924, 2015.

[40] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, Maio 2009.

[41] J. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, page 282–289. Morgan Kaufmann Publishers Inc., 2001.

[42] R. L. Stratonovich. Conditional markov processes. *Theory of Probability Its Applications*, 5(2): 156–178, 1960.

[43] A. McCallum, D. Freitag, and F. C. N. Pereira. Maximum entropy markov models for information extraction and segmentation. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, page 591–598. Morgan Kaufmann Publishers Inc., 2000.

[44] T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[45] S. Y. Wang, A. Quattoni, L. P. Morency, D. Demirdjian, and T. Darrell. Hidden conditional random fields for gesture recognition. In *CVPR '06: Proceedings of the 2006, IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1521–1527. IEEE, Janeiro 2006.

[46] C. Sutton and A. McCallum. An introduction to conditional random fields. *Foundations and Trends in Machine Learning*, 4(4):267–373, Abril 2012.

[47] A. Y. Ng and M. I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in Neural Information Processing Systems*, volume 14 of *NIPS 2001*, pages 841–848. Massachusetts Institute of Technology Press, Abril 2002.

[48] K. Q. Weinberger and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(9):207–244, Junho 2009.

[49] A. Gunawardana, M. Mahajan, A. Acero, and J. Platt. Hidden conditional random fields for phone classification. In *International Conference on Speech Communication and Technology*, Interspeech 2005. International Speech Communication Association, Setembro 2005.

[50] S. B. Imandoust and M. Bolandraftar. Application of k-nearest neighbor (knn) approach for predicting economic events: Theoretical background. *International Journal of Engineering Research and Applications*, 3(5):605–610, Outubro 2013.

[51] J. Walters-Williams and Y. Li. Comparative study of distance functions for nearest neighbors. In *Advanced Techniques in Computing Sciences and Software Engineering*, pages 79–84. Springer, 2010.

[52] J. G. Cleary. Analysis of an algorithm for finding nearest neighbors in euclidean space. *ACM Transactions on Mathematmal Software*, 5(2):183–192, Junho 1979.

[53] R. A. Melter. Some characterizations of city block distance. *Pattern Recognition Letters*, 6(4): 235–240, Setembro 1987.

[54] F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, volume 1 of *NAACL '03*, page 134–141. Association for Computational Linguistics, Junho 2003.

[55] L. P. Morency, C. M. Christoudias, A. Quattoni, H. Salamin, G. Stratou, and S. Wang. *Hidden-state Conditional Random Field Library - User Guide*, Janeiro 2010. Versão 2.0a.

[56] H. M. Wallach. Conditional random fields: An introduction. *Technical Reports (CIS)*, Fevereiro 2004. Relatório nº MS-CIS-04-21.

[57] A. Quattoni, S. Wang, L. P. Morency, M. Collins, T. Darrell, and M. Csail. Hidden-state conditional random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(10):1848–1852, 2007.

[58] J. F. D. Saa and M. Çetin. A latent discriminative model-based approach for classification of imaginary motor tasks from eeg data. *Journal of Neural Engineering*, 9(2), Março 2012.

[59] HCRF Library. `https://sourceforge.net/projects/hcrf/`, . Acedido a 2020-07-01.

[60] J. D. Rodriguez, A. Perez, and J. A. Lozano. Sensitivity analysis of k-fold cross validation in prediction error estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(3): 569–575, 2009.

[61] J. Shelton and G. P. Kumar. Comparison between auditory and visual simple reaction times. *Neuroscience and Medicine*, 1(1):30–32, 2010.

Anexo A

Contagem dos *feedbacks*

Tabela A.1: Contagem dos *feedbacks* motivacionais do jogo Toboggan.

***Feedback* motivacional - Toboggan**	**Contagem**
Vamos lá Vamos Vá lá	24
Calma Vamos lá aqui com calma	6
Bora Bora lá Vamos embora	24
Força	1
Está a ver o carro a tremer	1
Onde é que estão as bananas?	32
Consegue ver as bananas? Está a ver a banana?	33
Procure lá Procure lá as bananas Vamos procurar as bananas	2
Olhe a banana Vai apanhar aquela	5
Apanhou Conseguiu Ainda apanhou Ainda apanhámos Ainda conseguiu apanhar	12
Essa foi para o saco Mais um cacho Mais uma	4

Tabela A.1 - continuação da página anterior

Foi por um triz	1
Vamos apanhá-las Não as deixe fugir Vai deixar fugir as bananas Vamos apanhar as bananas Vai apanhar Vá apanhar Vai apanhá-las	13
Apanhámos uma banana Já está	3
Está a ir bem Está a correr bem Estamos num bom caminho	8
Boa velocidade	1
Muito bem	25
Está a apanhar as bananas (quase) todas Não deixa escapar nenhuma Farta de apanhar bananas Você está a apanhá-las	5
Está a ir muito rápido, a 100% Vamos rápido Vai a abrir	4
Isso mesmo	19
Exactamente Exacto	15
Ainda tem que apanhar muita banana Tem que apanhar mais bananas	2
Está bem assim	3
Já estamos a andar Estamos a ir	6
Bom controlo	4
Agora é que é sempre a abrir A todo o gás agora	2
Não faz mal Deixe estar Não se preocupe	13
Vem outra Vem aí outra Vem aí umas Vem aí mais Tem aí mais uma	11

Tabela A.1 - continuação da página anterior

Há mais	4
Outra vez Vamos tentar outra vez	5
Continue Continua a andar Continue no verde Segue	9
Frente frente	3
Ainda falta um bocadinho Está a acabar o tempo Não pára, ainda não parou o jogo	3
Acelera acelera Acelere, ainda apanha aquelas	5
Boa	114
Isso	64

Tabela A.2: Contagem dos *feedbacks* motivacionais do jogo Grape Stomping.

***Feedback* motivacional - Grape Stomping**	**Contagem**
Não faz mal, faz o que conseguir Faz aquilo que consegue Vai ser capaz Calma, elas vão-se pisando Agora vai correr melhor	7
Bora Bora lá	163
Outra vez	2
Você consegue Não deixe que o cansaço vença	11
Vamos lá Vá lá Vamos lá tentar Vamos embora Vamos ao trabalho Vamos continuar	181
Comece Comece a pisar Vamos lá começar	4
Não desiste Não pára Já não abranda Não pode desistir	18

Tabela A.2 - continuação da página anterior

Continua/Continue Continue a pisar Continue nesse ritmo	75
Isso mesmo É assim mesmo	42
Levanta Levanta e esmaga Levanta e pisa Sobe e desce Levanta um, levanta outro	39
Já está Já vai pisando	9
Marchar	2
Mais para cima/Mais Bem para cima os joelhos Pé para cima Levante bem a perna Sobe mais/ Mais alto Puxe a pernoca para cima Mais o pé/ Mais o joelho Levantar bem os joelhos Esmagar bem	59
Muito bem Óptimo Excelente	36
Pisa Pisa a uva Pise as uvas Vamos pisar	47
Joelhos para cima Levantar os joelhos Para cima Sobe Pernas para cima	67
Mais velocidade Mais rápido Joelho rápido (que é para pisar mais uvas) Pisar mais Com mais rapidez	12
Mais força Com firmeza Mais força no chão Pisa com força Pise bem	15

Tabela A.2 - continuação da página anterior

Tantos verdes Já está mais verde que vermelho Está a ficar verde	3
Dobra Dobre o joelho Dobre e põe no chão Dobra bem o joelho para cima Dobrar mais	46
Pisar as uvas Tem que pisar as uvas Temos muitas uvas para pisar Sempre a pisar agora	11
Tente lá Faça lá	3
Conseguir mais um barril	3
Já enchemos mais um Já encheu um balde Mais um balde cheio	7
1, 2, 3, 4	18
Pisa as que aí estão Pisa tudo Pise lá essas Ainda falta pisar aquelas Tem muita uva aí agora Está a ficar com muitas uvas por pisar Está a ficar com o cesto cheio de uvas Ainda há uvas Ainda estão aí uvas Aquela ainda está inteira Acabe de pisar essas Ainda faltam mais uns cachos	31
Já temos o balde praticamente cheio Estamos quase a encher o barril/outro barril	4
Está a correr (muito) bem Está a fazer bem Está a ir bem Já estamos a pisar muito bem Agora está a ir Está a ver como consegue Está a conseguir Está a pisar bastantes uvas	32
Bom esforço Bom trabalho	3

Tabela A.2 - continuação da página anterior

Está melhor Está óptimo Agora está a fazer bem Agora está a correr bem Agora já está a conseguir Já está a pisar bastantes uvas	8
Temos que fazer muitos barris	1
Vêm aí mais uvas Vem aí a uva Vem aí mais um cacho E mais outro Mais um cacho de uvas Aí vem elas	12
Não perca a força Mais genica	3
Força	43
Direita esquerda Um e outro	38
Já lhe apanhou o jeito Já percebeu como é Já está a pisar as uvas Agora está a pisar todas	5
Bom ritmo Boa passada	3
Está quase Está quase a acabar o tempo Estamos quase a terminar Já falta pouco Ainda não acabou Estamos quase a chegar ao fim Temos de pisar um bocadinho mais Faltam poucas uvas Estamos na recta final Mais um bocadinho Tente lá mais um bocadinho	42
Exactamente Exacto	14
Boa Que bom Bem	164
Isso	152
Devagarinho	1

Tabela A.3: Contagem dos *feedbacks* motivacionais do jogo Rabelos.

***Feedback* motivacional - Rabelos**	**Contagem**
Pode começar Comece Vamos começar	7
Bora Bora lá Vamos embora	134
Continue Continua a remar Continue em frente	171
Outra vez	3
Reme/Rema Remar Rodar Reme bem Temos que remar Tem que remar	51
Não pára de remar Não pára	11
Não faz mal	5
Está a ver a barra verde? Olhe a barra verde Viu o pontinho verde?	2
Já temos o barco a andar Está a andar Já estamos a andar	2
Verde Continue no verde Siga o verde	8
Fugiram Fugiram de nós	3
Toque aqui Toca	2
Frente E vai à frente Ir à frente Para a frente Reme para a frente	41
Mais um barril Outro	5
Já está	3

Tabela A.3 - continuação da página anterior

Mais um bocadinho É só mais um bocadinho Ainda não acabámos	17
Vamos lá Vá lá Vamos remar Vamos a isso	211
Vamos acelerar	2
Está mais rápido Está a acelerar a remada	2
Isso mesmo É isso mesmo É isso Assim é que é	51
Muito bem	40
Está excelente Está óptimo	8
Está a remar bem Está a ir muito bem Está a fazer muito bem Está no verde está bem Assim vamos longe Gosto do gesto Está a conseguir Já apanhou o jeito	30
Bom ritmo Agora já estamos a bom andar Vamos andar a bom andar Vai a um bom ritmo Sempre certinho	11
Mantenha a posição Mantenha o remar É esse o movimento É esse o gesto Boa remada Continue assim	7
Exactamente Exacto	13
Força Com genica	20
Apanhou	2
A toda a velocidade agora Tudo por tudo agora	4

Tabela A.3 - continuação da página anterior

Não desiste	13
Você consegue	3
Estamos na recta final Estamos a chegar Está quase Já passámos o meio Está quase a acabar o tempo Já falta pouco Estamos quase a chegar ao fim Já não falta tudo	43
Movimento dos braços Braços Ombros	12
Tronco	2
Apanhou os barris todos Já foram todos	3
Temos aqui mais barris Faltam mais barris	3
Vamos buscar estes Vamos apanhar os barris	2
Acelere Mais rápido Consegue fazer um bocadinho mais rápido agora	3
Boa	152
Isso	116

Tabela A.4: Contagem dos *feedbacks* motivacionais do jogo ExerPong.

***Feedback* motivacional - ExerPong**	**Contagem**
Força	1
Vamos começar	3
Bom trabalho Ótimo	3
Está tão pequena a barra Está tão pequenina Temos a barra muito pequena	4
Vamos lá Vamos Vamos a outra Vá lá Vamos embora	98

Tabela A.4 - continuação da página anterior

Isso mesmo É isso mesmo É assim mesmo Isso é que é	33
Exatamente Exato	22
Muito bem	50
Excelente	12
Consegue ver a bola? Onde está a bola?	4
Onde está a barra? Está a ver a barra?	3
Mexa a barra, mexa a barra Mexa lá Mexa lá a barra	5
Não deixe cair a bola Não pode deixar cair a bola Não deixe cair Olhe que ela vai cair	32
Vá apanhá-la Olha essa	3
Está a ir muito bem Está a correr muito bem Já está a controlar melhor a barra Está muito bom Está a conseguir	5
Concentração Concentre-se	2
Está a fazer bem	1
É esse o objectivo	1
Não pára, não desiste Não desanime Tentar não deixar cair agora até ao fim	6
Estava quase Estava lá quase Quase Boa tentativa Por pouco Foi quase	9
Foi bom Essa foi muito boa	2
Já derrubou muitos quadrados	1

Tabela A.4 - continuação da página anterior

Está quase Estamos mesmo a terminar Já falta pouco tempo Ainda falta um bocadinho	9
Foi na pontinha Foi à tangente Mesmo na pontinha Ainda deu Por um triz Foi à pontinha mas foi	6
Bora Bora lá	35
Continua Vamos continuar	14
Isto não pára, está sempre a andar	1
Já está	9
A bola continua a rodar Não parou, a bola segue	4
Controla	4
Devagar Devagarinho	5
Tente lá mais um bocadinho Ainda faltam aqueles quadrados	3
Há mais bolas	1
Não faz mal Não se preocupe	26
Vem aí outra Ainda vem outra vez	9
Vamos lá para a próxima Vai apanhar mais	2
Outra vez Novamente	13
Boa jogada Boa antecipação Jogada de mestre Boa coordenação	4
Boa	182
Isso	76

Tabela A.5: Contagem dos *feedbacks* motivacionais do jogo ExerFado.

***Feedback* motivacional - ExerFado**	**Contagem**
Vai começar	1
Pronto	3
Exactamente Exacto	8
Onde está a nota a cair? Onde é que ela está? Está a ver aí a nota? Está a ver a nota a descer? Então a nota? Qual é a nota? Consegue ver as notas a descerem o piano?	16
Consegue ver a bola azul?	1
Está a ver a X a descer? (X = cor específica da nota)	3
E agora está alguma a descer? E agora não há nenhuma? E agora?	16
Está a ver Está a ver ali Está a ver aquela	11
Chegue	4
Bora	15
Não faz mal Não tem problema	6
Vamos continuar Continue	2
Já está	2
Vamos lá Vá lá	4
Não deixe cair Apanhe	5
Vamos a esta Vamos para lá	2
É mais difícil Está mais difícil É mais difícil	5
Já conseguiu apanhar	1
Está a conseguir	1
É mesmo assim	1

Tabela A.5 - continuação da página anterior

Vai conseguir	2
Muito bem Espectáculo	20
Tente lá pressionar a tecla	1
Consegue perceber quando a tecla está pressionada?	1
Está ali outra Mais outra Lá vem outra	3
Estamos quase a chegar ao fim	1
Afasta os pés e junta Afasta junta	3
Mas correu bem	1
Isso mesmo É isso mesmo	11
Boa	51
Isso	28

Anexo B

Tabelas com as *accuracies* das *labels*

Neste anexo, constam todas as tabelas associadas às *labels*, apresentando não só a média das *accuracies* de cada uma, como também os respectivos valores de desvio padrão.

Tabela B.1: Resultados das *accuracies* das *labels*, com o código KNN_algorithm.m, para os vários ficheiros indicados.

Intervalos de tempo		**Média da *accuracy* de cada *label***							**Desvio padrão da *accuracy* de cada *label***						
Antes (ms)	**Depois (ms)**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
200	200	0.1844	0.1625	0.0685	0.3020	0.0165	0.1932	0.1444	0.0136	0.0559	0.0225	0.0447	0.0144	0.0674	0.0803
250	250	0.1708	0.1579	0.0656	0.3115	0.0191	0.1828	0.1481	0.0082	0.0658	0.0218	0.0515	0.0145	0.0542	0.0892
300	300	0.1599	0.1498	0.0786	0.3092	0.0210	0.1900	0.1578	0.0076	0.0630	0.0075	0.0623	0.0107	0.0556	0.0937
350	350	0.1553	0.1499	0.0642	0.2967	0.0292	0.1805	0.1590	0.0130	0.0511	0.0330	0.0290	0.0264	0.0553	0.0957
500	500	0.1701	0.1441	0.0622	0.2945	0.0565	0.1804	0.1451	0.0336	0.0416	0.0086	0.0471	0.0379	0.0432	0.0774
1000	1000	0.1440	0.1456	0.0553	0.3246	0.0325	0.2125	0.1378	0.0307	0.0350	0.0091	0.0528	0.0280	0.0143	0.0673
1500	1500	0.1298	0.1622	0.0491	0.3234	0.0423	0.2092	0.1282	0.0200	0.0301	0.0082	0.0275	0.0586	0.0324	0.0512
2000	2000	0.1385	0.1481	0.0441	0.3291	0.0375	0.2142	0.1385	0.0387	0.0467	0.0110	0.0288	0.0512	0.0314	0.0420
200	0	0.1642	0.1833	0.0383	0.3080	0.0348	0.1853	0.1266	0.0509	0.0911	0.0100	0.0632	0.0060	0.0423	0.0781
250	0	0.1314	0.1609	0.0366	0.2954	0.0221	0.1702	0.1355	0.0278	0.0701	0.0091	0.0680	0.0155	0.0313	0.0956
300	0	0.1331	0.1714	0.0626	0.3005	0.0311	0.1854	0.1438	0.0462	0.0640	0.0321	0.0804	0.0107	0.0320	0.1070
350	0	0.1259	0.1780	0.0643	0.2904	0.0322	0.1832	0.1565	0.0509	0.0589	0.0393	0.0612	0.0115	0.0364	0.1269
500	0	0.1005	0.1686	0.0560	0.3023	0.0354	0.1878	0.1740	0.0397	0.0307	0.0103	0.0545	0.0267	0.0295	0.1166
1000	0	0.1381	0.1390	0.0470	0.3186	0.0251	0.2082	0.1823	0.0416	0.0247	0.0147	0.0794	0.0108	0.0258	0.1028
1500	0	0.1365	0.1451	0.0456	0.3069	0.0287	0.1991	0.1670	0.0346	0.0185	0.0235	0.0553	0.0219	0.0408	0.1032
2000	0	0.1273	0.1536	0.0455	0.2897	0.0373	0.1960	0.1615	0.0380	0.0212	0.0266	0.0694	0.0187	0.0451	0.0916
500	300	0.1490	0.1480	0.0657	0.2930	0.0512	0.1929	0.1662	0.0191	0.0435	0.0067	0.0571	0.0279	0.0528	0.0950
750	300	0.1433	0.1366	0.0688	0.3119	0.0515	0.1952	0.1595	0.0281	0.0517	0.0050	0.0639	0.0235	0.0451	0.0826
1000	300	0.1433	0.1339	0.0585	0.3197	0.0322	0.2097	0.1582	0.0310	0.0368	0.0049	0.0619	0.0246	0.0446	0.0858
1500	300	0.1544	0.1394	0.0483	0.3098	0.0370	0.2048	0.1582	0.0383	0.0204	0.0225	0.0455	0.0271	0.0434	0.0991

Tabela B.2: Resultados das *accuracies* das *labels*, com o código CRF_algorithm.m, para os vários ficheiros indicados com o parâmetro *windowSize* = 0.

Intervalos de tempo		**Média da *accuracy* de cada *label***							**Desvio padrão da *accuracy* de cada *label***						
Antes (ms)	**Depois (ms)**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
200	200	0.0968	0.0415	0.1344	0.3803	0.1012	0.1925	0.1650	0.1086	0.0096	0.0314	0.0583	0.0632	0.1513	0.1204
250	250	0.1060	0.0477	0.2353	0.3874	0.0959	0.1906	0.1492	0.0859	0.0251	0.0947	0.0944	0.0803	0.1445	0.0955
300	300	0.1333	0.0475	0.2876	0.3633	0.0973	0.2119	0.1205	0.1007	0.0322	0.1199	0.1041	0.0730	0.1472	0.0899
350	350	0.1418	0.0505	0.2239	0.3590	0.1390	0.2054	0.0943	0.1128	0.0214	0.1774	0.1403	0.1448	0.1173	0.0433
500	500	0.2219	0.0876	0.1649	0.3398	0.1053	0.1928	0.0921	0.1368	0.0549	0.2181	0.2014	0.1483	0.1138	0.0463
1000	1000	0.3007	0.1054	0.0300	0.3816	0.1559	0.1045	0.0637	0.2170	0.0150	0.0520	0.1749	0.0800	0.0917	0.0804
1500	1500	0.2435	0.0906	0.1461	0.3842	0.2145	0.0775	0.1100	0.1900	0.0222	0.1842	0.1552	0.2146	0.0304	0.0462
2000	2000	0.2810	0.0820	0.1071	0.3546	0.1942	0.1360	0.0649	0.1920	0.0231	0.1855	0.2259	0.1564	0.0698	0.0689
200	0	0.0742	0.0390	0.3348	0.3986	0.1624	0.1544	0.1505	0.0643	0.0270	0.3165	0.0916	0.1771	0.0988	0.1641
250	0	0.1261	0.0620	0.3408	0.3907	0.1363	0.1673	0.1719	0.1208	0.0318	0.3280	0.0624	0.1318	0.1281	0.1472
300	0	0.1462	0.0524	0.2896	0.3850	0.1103	0.1822	0.1552	0.1368	0.0162	0.2675	0.0767	0.0495	0.0873	0.1054
350	0	0.1885	0.0607	0.2777	0.3675	0.0981	0.1790	0.1187	0.1666	0.0233	0.3338	0.1149	0.0606	0.0281	0.0926
500	0	0.1480	0.0405	0.3014	0.3375	0.1604	0.2109	0.1568	0.1253	0.0337	0.3210	0.1788	0.0848	0.0688	0.1455
1000	0	0.2276	0.0771	0.0719	0.3379	0.1491	0.2318	0.1363	0.1577	0.0571	0.0854	0.1342	0.0787	0.0659	0.1183
1500	0	0.2406	0.0682	0.2316	0.3359	0.1484	0.1841	0.1619	0.1501	0.0483	0.3058	0.0846	0.1287	0.0323	0.1450
2000	0	0.2424	0.1049	0.0777	0.3825	0.1202	0.1888	0.1130	0.2052	0.0833	0.0886	0.1155	0.1095	0.0513	0.0772
500	300	0.1795	0.0405	0.2201	0.2984	0.0759	0.2200	0.1665	0.1078	0.0070	0.1709	0.1439	0.0925	0.1340	0.1288
750	300	0.2014	0.0800	0.1990	0.3034	0.0499	0.2071	0.1954	0.1764	0.0597	0.1885	0.1053	0.0344	0.1228	0.0673
1000	300	0.2108	0.0880	0.2137	0.2882	0.1308	0.2043	0.1333	0.1826	0.0651	0.3205	0.0885	0.1096	0.1023	0.0539
1500	300	0.2134	0.0768	0.1862	0.3070	0.1404	0.1600	0.1956	0.1535	0.0516	0.2009	0.1317	0.1216	0.0905	0.1556

Tabela B.3: Resultados das *accuracies* das *labels*, com o código CRF_algorithm.m, para os vários ficheiros indicados com o parâmetro *windowSize* = 1.

Intervalos de tempo		**Média da *accuracy* de cada *label***							**Desvio padrão da *accuracy* de cada *label***						
Antes (ms)	**Depois (ms)**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
200	200	0.0985	0.0727	0.0562	0.2996	0.1764	0.2932	0.1851	0.0531	0.0820	0.0576	0.0772	0.1846	0.1068	0.1195
250	250	0.1204	0.0703	0.1147	0.3288	0.1467	0.2366	0.1540	0.0561	0.0400	0.1196	0.1411	0.1493	0.0801	0.0456
300	300	0.1414	0.0569	0.1527	0.3587	0.1693	0.1928	0.1147	0.0944	0.0432	0.2133	0.1873	0.1993	0.1260	0.0509
350	350	0.1567	0.0808	0.1678	0.3470	0.2336	0.1866	0.0976	0.1322	0.0830	0.1968	0.1432	0.1582	0.1029	0.0345
500	500	0.1596	0.0915	0.0380	0.3617	0.2374	0.1929	0.1013	0.0785	0.0394	0.0293	0.2386	0.2527	0.1157	0.0340
1000	1000	0.1990	0.0721	0.0166	0.3863	0.1908	0.1669	0.0865	0.1262	0.0684	0.0208	0.0856	0.2256	0.0229	0.0524
1500	1500	0.2390	0.0790	0.0157	0.3840	0.1093	0.2458	0.0859	0.2003	0.0951	0.0150	0.0614	0.0857	0.0707	0.0035
2000	2000	0.1475	0.0784	0.0169	0.3283	0.1363	0.2484	0.1024	0.0959	0.0801	0.0241	0.0785	0.1278	0.0630	0.0753
200	0	0.1602	0.0529	0.2435	0.4065	0.1475	0.1704	0.1867	0.1332	0.0250	0.1527	0.1440	0.1686	0.0787	0.1101
250	0	0.1788	0.0538	0.2868	0.4139	0.1241	0.1927	0.1531	0.1564	0.0268	0.3111	0.0834	0.1587	0.0677	0.1041
300	0	0.1860	0.0459	0.1714	0.4047	0.1445	0.1995	0.1350	0.2153	0.0339	0.2144	0.1221	0.2327	0.0756	0.1309
350	0	0.2066	0.0496	0.2435	0.3991	0.1490	0.2037	0.1373	0.1848	0.0560	0.2849	0.1351	0.1878	0.0511	0.1299
500	0	0.1413	0.0651	0.2909	0.3593	0.1799	0.2373	0.0865	0.1159	0.0350	0.3277	0.2509	0.2607	0.0465	0.0553
1000	0	0.2375	0.0834	0.2027	0.3183	0.2074	0.2542	0.1477	0.1627	0.0741	0.2918	0.2270	0.1716	0.1315	0.0139
1500	0	0.2355	0.0853	0.2388	0.3683	0.2000	0.1302	0.1911	0.1638	0.0251	0.2546	0.0416	0.1991	0.0516	0.0943
2000	0	0.2434	0.1091	0.0808	0.4278	0.0654	0.2071	0.1352	0.1949	0.0359	0.0773	0.0142	0.0689	0.0230	0.1021
500	300	0.1763	0.0522	0.1492	0.3168	0.2359	0.2295	0.0865	0.0796	0.0525	0.2279	0.1336	0.2673	0.1467	0.0376
750	300	0.1783	0.0672	0.1561	0.2807	0.0973	0.2636	0.1165	0.0938	0.0321	0.2388	0.1726	0.0724	0.1977	0.0524
1000	300	0.2362	0.1029	0.1639	0.2773	0.1256	0.2630	0.1993	0.1267	0.1073	0.2370	0.1760	0.1052	0.1254	0.0418
1500	300	0.2090	0.1057	0.1417	0.3603	0.1287	0.2168	0.1100	0.1940	0.0603	0.1765	0.0369	0.0796	0.0398	0.0514

Tabela B.4: Resultados das *accuracies* das *labels*, com o código CRF_algorithm.m, para os vários ficheiros indicados com o parâmetro *windowSize* = 3.

Intervalos de tempo		**Média da *accuracy* de cada *label***							**Desvio padrão da *accuracy* de cada *label***						
Antes (ms)	**Depois (ms)**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
200	200	0.1129	0.0658	0.0534	0.3450	0.1807	0.2479	0.2270	0.0253	0.0560	0.0554	0.1154	0.1835	0.1048	0.0805
250	250	0.1474	0.0746	0.1381	0.3598	0.1003	0.2381	0.1335	0.0851	0.0596	0.1057	0.1617	0.0737	0.1145	0.0274
300	300	0.1460	0.0792	0.1302	0.3913	0.0690	0.2859	0.1172	0.0487	0.0957	0.1575	0.1797	0.0994	0.1071	0.0484
350	350	0.1597	0.0997	0.0464	0.3771	0.1313	0.1902	0.1527	0.0896	0.0932	0.0299	0.1247	0.0593	0.1101	0.0048
500	500	0.1519	0.0665	0.0625	0.3628	0.1777	0.2581	0.0891	0.0811	0.0402	0.0279	0.1705	0.1193	0.1794	0.0345
1000	1000	0.1942	0.1466	0.0227	0.3747	0.0727	0.2105	0.0951	0.2036	0.0551	0.0208	0.1795	0.0526	0.1254	0.0716
1500	1500	0.1865	0.1515	0.0124	0.4886	0.0198	0.2117	0.0369	0.1435	0.0994	0.0073	0.1151	0.0310	0.0498	0.0443
2000	2000	0.1747	0.1140	0.0439	0.3663	0	0.2059	0.0607	0.1733	0.0360	0.0617	0.0155	0	0.0758	0.0343
200	0	0.1581	0.0628	0.2144	0.3610	0.0916	0.1623	0.1869	0.1264	0.0442	0.1699	0.1117	0.1225	0.0182	0.0414
250	0	0.1637	0.0803	0.1737	0.3747	0.1193	0.1912	0.1373	0.1346	0.0475	0.1982	0.1446	0.1716	0.0333	0.0540
300	0	0.1604	0.0378	0.2170	0.3567	0.1737	0.1875	0.1401	0.1340	0.0382	0.3705	0.1440	0.1556	0.0336	0.1225
350	0	0.1180	0.0867	0.1867	0.3760	0.1914	0.1892	0.1665	0.1111	0.0650	0.2788	0.1875	0.1931	0.0704	0.1325
500	0	0.0894	0.0538	0.2721	0.4358	0.1645	0.1979	0.1092	0.0999	0.0418	0.3070	0.1934	0.1267	0.0504	0.0573
1000	0	0.2335	0.0595	0.1396	0.3525	0.1602	0.2346	0.1021	0.1285	0.0615	0.0719	0.1978	0.0970	0.1051	0.0543
1500	0	0.0982	0.0617	0.0875	0.3551	0.1225	0.2090	0.1820	0.0612	0.0344	0.0544	0.1197	0.0751	0.0866	0.0648
2000	0	0.2070	0.1329	0.0160	0.3343	0.0251	0.2832	0.1057	0.1911	0.0902	0.0266	0.1854	0.0231	0.1315	0.0749
500	300	0.1573	0.0896	0.0261	0.4270	0.1335	0.2457	0.0781	0.0278	0.1198	0.0309	0.1674	0.0760	0.0981	0.0194
750	300	0.1390	0.0531	0.0603	0.3314	0.1534	0.2747	0.1064	0.0623	0.0342	0.0524	0.1954	0.1460	0.0934	0.0898
1000	300	0.0970	0.1073	0.0655	0.3174	0.0445	0.2675	0.1060	0.1037	0.1116	0.0500	0.1875	0.0306	0.1356	0.0645
1500	300	0.2152	0.1282	0.0304	0.2860	0.0210	0.3283	0.0923	0.2109	0.0396	0.0290	0.1593	0.0309	0.1096	0.0147

Tabela B.5: Resultados das *accuracies* das *labels*, com o código HCRF_algorithm.m, para o ficheiro [300,300]ms.

Parâmetros [300,300]ms		**Média da *accuracy* de cada *label***							**Desvio padrão da *accuracy* de cada *label***						
nbHiddenStates	***windowSize***	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	1	0.0237	0	0	1	0	0	0	0.0031	0	0	0	0	0	0
	3	0.0237	0	0	1	0	0	0	0.0031	0	0	0	0	0	0
	5	0.0237	0	0	1	0	0	0	0.0031	0	0	0	0	0	0
2	1	0.1560	0.0445	0	0.7484	0	0.0359	0	0.1123	0.0455	0	0.1837	0	0.0320	0
	3	0.1009	0.0426	0.0556	0.8003	0	0.0408	0	0.0739	0.0737	0.0962	0.3071	0	0.0707	0
	5	0.0407	0.2345	0	0.7391	0	0	0	0.0400	0.3867	0	0.4182	0	0	0
3	1	0.1434	0.2559	0.0392	0.6226	0	0.0306	0.0418	0.0746	0.3866	0.0679	0.3680	0	0.0368	0.0364
	3	0.1279	0.0828	0	0.7159	0	0.1901	0.0098	0.1779	0.1255	0	0.1989	0	0.1359	0.0170
	5	0.1050	0.2328	0	0.6535	0	0.0434	0	0.0735	0.1944	0	0.3055	0	0.0447	0
5	1	0.0711	0.2364	0.0196	0.5296	0.0238	0.2792	0.0381	0.0094	0.2682	0.0340	0.2816	0.0412	0.3477	0.0188
	3	0.0727	0.1723	0.0551	0.4322	0.0256	0.1950	0.1362	0.0649	0.1029	0.0592	0.3462	0.0444	0.1087	0.1878
	5	0.0531	0.3143	0	0.5339	0	0.0315	0.0148	0.0229	0.4060	0	0.4765	0	0.0280	0.0257

Tabela B.6: Resultados das *accuracies* das *labels*, com o código HCRF_algorithm.m, para o ficheiro [500,300]ms.

Parâmetros [500,300]ms		**Média da *accuracy* de cada *label***							**Desvio padrão da *accuracy* de cada *label***						
nbHiddenStates	***windowSize***	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	1	0.0237	0	0	1	0	0	0	0.0031	0	0	0	0	0	0
	3	0.0237	0	0	1	0	0	0	0.0031	0	0	0	0	0	0
	5	0.0237	0	0	1	0	0	0	0.0031	0	0	0	0	0	0
2	1	0.1252	0.2456	0.0784	0.6461	0	0.1008	0.0405	0.1023	0.2850	0.1358	0.1148	0	0.0346	0.0446
	3	0.1393	0.3135	0	0.5595	0	0.0238	0.0490	0.1156	0.4852	0	0.4666	0	0.0412	0.0849
	5	0.0837	0.2344	0.0196	0.5814	0.0256	0.0465	0.0778	0.0263	0.3877	0.0340	0.2695	0.0444	0.0806	0.1347
3	1	0.1001	0.1017	0	0.5796	0	0.2558	0.0074	0.0563	0.0963	0	0.3917	0	0.4431	0.0128
	3	0.1055	0.0758	0.0317	0.6533	0	0.1840	0.0196	0.0830	0.1312	0.0550	0.4307	0	0.2606	0.0340
	5	0.1591	0.0600	0.0196	0.6176	0	0.1089	0.0307	0.1014	0.0849	0.0340	0.2610	0	0.1690	0.0295
5	1	0.1535	0.3405	0	0.4316	0	0.0622	0.0593	0.1313	0.3803	0	0.3332	0	0.0883	0.0559
	3	0.1046	0.3950	0	0.5362	0.0185	0.0476	0	0.0572	0.4104	0	0.3940	0.0321	0.0825	0
	5	0.1553	0.2317	0.1064	0.3035	0.0256	0.1766	0.1492	0.0361	0.0158	0.0937	0.1253	0.0444	0.0795	0.1314

Tabela B.7: Resultados das *accuracies* das *labels*, com o código HCRF_algorithm_final.m, para o ficheiro [300,300]ms.

Parâmetros [300,300]ms		**Média da *accuracy* de cada *label***							**Desvio padrão da *accuracy* de cada *label***						
nbHiddenStates	***windowSize***	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	1	0.0237	0.6667	0	0.3333	0	0	0	0.0031	0.5774	0	0.5774	0	0	0
	3	0.0237	0.6667	0	0	0	0.3333	0	0.0031	0.5774	0	0	0	0.5774	0
	5	0.0237	0	0	0.3333	0	0.3333	0.3333	0.0031	0	0	0.5774	0	0.5774	0.5774
2	1	0.0757	0.0984	0.2813	0.1917	0.1007	0.0779	0.1739	0.0754	0.1026	0.1813	0.1778	0.1181	0.0404	0.0760
	3	0.0688	0.0071	0.3221	0.1986	0.0238	0.3163	0.2342	0.0809	0.0123	0.4401	0.2954	0.0412	0.4873	0.3576
	5	0.0392	0.0833	0.1111	0.0586	0	0.3377	0.3229	0.0266	0.1443	0.1925	0.0370	0	0.5035	0.3744
3	1	0.0904	0.1452	0	0.1080	0.6443	0.0298	0	0.0627	0.2333	0	0.1428	0.2710	0.0515	0
	3	0.1877	0.0227	0.2178	0.2422	0.2816	0.1031	0.0098	0.1022	0.0394	0.0885	0.1254	0.1821	0.1349	0.0170
	5	0.0678	0.1721	0.0747	0.3347	0.1467	0.1004	0.0641	0.0356	0.1789	0.0913	0.3601	0.2079	0.0881	0.0668
5	1	0.0535	0.0171	0.3732	0.0437	0	0.5111	0	0.0300	0.0151	0.4919	0.0379	0	0.4481	0
	3	0.1770	0.0914	0.0868	0.1568	0.2436	0.0923	0.3259	0.1384	0.0501	0.0763	0.1721	0.2502	0.0830	0.5079
	5	0.2375	0.1266	0.1109	0.1542	0.2261	0.0623	0.0749	0.0561	0.0744	0.0700	0.1796	0.1566	0.0540	0.0627

Tabela B.8: Resultados das *accuracies* das *labels*, com o código HCRF_algorithm_final.m, para o ficheiro [500,300]ms.

Parâmetros [500,300]ms		**Média da *accuracy* de cada *label***							**Desvio padrão da *accuracy* de cada *label***						
nbHiddenStates	***windowSize***	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	1	0.0237	0.3333	0	0.3333	0	0.3333	0	0.0031	0.5774	0	0.5774	0	0.5774	0
	3	0.0237	0	0	0	0.6667	0.3333	0	0.0031	0	0	0	0.5774	0.5774	0
	5	0.0237	0	0	0.6667	0.3333	0	0	0.0031	0	0	0.5774	0.5774	0	0
2	1	0.1347	0.3597	0.0317	0.2363	0	0.1463	0.0492	0.1482	0.5096	0.0550	0.1875	0	0.1268	0.0452
	3	0.0747	0.0152	0.0159	0.0175	0.3297	0.2874	0.2889	0.0417	0.0262	0.0275	0.0304	0.4530	0.4629	0.5004
	5	0.0380	0.0286	0.0784	0.4206	0.0698	0.0775	0.3597	0.0488	0.0495	0.1358	0.3708	0.0779	0.1343	0.4036
3	1	0.0609	0.0219	0.1298	0.1745	0.4560	0.1264	0.1451	0.0330	0.0214	0.1501	0.1360	0.2828	0.1746	0.0849
	3	0.1030	0.0303	0.3417	0.0580	0.0556	0.0945	0.4301	0.0913	0.0525	0.4736	0.0508	0.0962	0.0785	0.4594
	5	0.0773	0.1509	0.3604	0.2215	0.3152	0.0510	0.0407	0.0290	0.1882	0.4796	0.3210	0.2255	0.0445	0.0525
5	1	0.0805	0.0906	0.3408	0.1211	0.2253	0.1777	0.0407	0.0594	0.0636	0.4919	0.0956	0.2310	0.1914	0.0525
	3	0.0544	0.0071	0.2698	0.1360	0.2381	0.0357	0.3627	0.0080	0.0123	0.4674	0.1417	0.4124	0.0619	0.5536
	5	0.1350	0.0880	0.0196	0.0960	0.0513	0.3109	0.3865	0.1199	0.1345	0.0340	0.0897	0.0888	0.4416	0.5110