INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Área Departamental de Engenharia Electrónica e Telecomunicações e de Computadores

ISEL

Reconhecimento de Língua Gestual

André de Oliveira Monteiro Castanheira Costa
(Licenciado em Engenharia Informática e Computadores)

Trabalho Final de Mestrado para obtenção do grau de Mestre
em Engenharia Informática e Computadores

Orientadores:
Doutor Pedro Miguel Torres Mendes Jorge
Mestre Lara Lourenço Santos

Júri:

Presidente:
Doutor Fernando Manuel Gomes de Sousa

Vogais:
Doutor Artur Jorge Ferreira
Doutor Pedro Miguel Torres Mendes Jorge

Setembro de 2014

Resumo

As Línguas Gestuais são o canal privilegiado para a comunicação de quem sofre de incapacidade auditiva total ou parcial. Devido ao facto destas Línguas Gestuais terem surgido por força da necessidade de comunicação de grupos de indivíduos por todo o mundo levou a que pessoas de regiões distintas utilizassem gestos diferentes para as mesmas palavras. Ao longo dos anos a comunidade científica tem tentado desenvolver sistemas que tentam automatizar o reconhecimento destas línguas. Devido à complexidade intrínseca das línguas, pela diversidade física dos executores ou pelas limitações tecnológicas, até aos dias de hoje, não é conhecida nenhuma solução óptima ou que resolva o problema na íntegra. Com o surgir e o aperfeiçoamento de dispositivos sensoriais de profundidade, surgem novas possibilidades para abordar o problema. Nesta tese, foram definidas, desenvolvidas e testadas propostas de soluções que se acredita serem uma mais-valia para o problema do reconhecimento automático de gestos aplicado à Língua Gestual Portuguesa. Para o efeito, foi adoptado um sensor de profundidade para a aquisição de gestos naturais do ser humano, adoptadas técnicas de extracção de características e implementados algoritmos de classificação para resolver o problema do reconhecimento de gestos estáticos e dinâmicos de forma contínua em tempo real.

Palavras-chave

Reconhecimento de Língua Gestual Portuguesa, Sensores de Profundidade, Aprendizagem Automática, Template Matching, Reconhecimento Automático de Gestos, Análise de Componentes Principais, Alinhamento Temporal Dinâmico, Modelos de Markov Não-Observáveis

Abstract

The sign languages are a privileged channel for communication for those who suffer from total or partial hearing loss. Because these signs languages were developed by the need of communication of groups of individuals throughout the world, people from different regions use different gestures for the same words. Over the years the scientific community has tried to develop systems that attempt to automate the recognition of these languages. Due to the intrinsic complexity of the language, the physical diversity of the users or technological limitations, to this day, it is not known an optimal solution that totally solves this problem. With the rise and improvement of depth sensory devices, new possibilities emerge to address the problem. In this thesis, proposals were defined, developed and tested for solutions that we believe will be an asset to the problem of automatic recognition gestures for the Portuguese Sign Language. To this end, a depth sensor device for acquiring natural stimuli of the human being has been adapted, featuring extraction techniques and classification algorithms were implemented to solve the problem of the recognition of static and dynamic gestures in a continuous form and in real time.

Keywords

Portuguese Sign Language Recognition, Depth Sensor, Machine Leaarning, Template Matching, Automatic Gesture Recognition, Principal Component Analysis, Dynamic Time Warping, Hidden Markov Models.

Agradecimentos

Em primeiro lugar quero agradecer aos meus Pais pelo apoio incondicional e pela força que sempre me deram ao longo destes anos. Sem eles nada disto teria sido possível nem me teria tornado na pessoa que sou hoje.

Quero agradecer aos meus orientadores por terem aceite este desafio e por me terem dado a motivação e ajuda necessária para que este trabalho pudesse ser concluído.

Quero também agradecer aos meus colegas e amigos, destacando o meu amigo Pedro Miguel Completo Ferrerinha e o meu colega Ricardo Maranho, pelo companheirismo e pelas inúmeras horas de trabalho em conjunto.

Por último, não posso deixar de agradecer à minha entidade empregadora, pela compreensão nas diversas alturas de maior dificuldade a nível académico.

Índice

Índice de figuras

Índice de código

Índice de tabelas

Lista de Acrónimos

API	*Application Programming Interface*
ASL	*American Sign Language*
DTW	*Dynamic Time Warping*
FPS	*Frames Per Second*
HCI	*Human-Computer Interaction*
HMM	*Hidden Markov Models*
I/O	*Input/Output*
IR	Infra-Red
LED	*Light Emitting Diode*
LGP	Língua Gestual Portuguesa
NN	*Neural Network*
NUI	*Natural User Interfaces*
PCA	*Principal Component Analysis*
PUI	*Perceptual User Interface*
RGB	*Red Green and Blue*
SDK	*Software Development Kit*
TM	*Template Matching*
USB	*Universal Serial Bus*
VGA	*Video Graphics Array*

1. Introdução

Os gestos fazem parte do dia-a-dia de qualquer indivíduo no decorrer do processo de comunicação e são executados muitas vezes de forma inconsciente para exprimir uma ideia ou enfatizar o diálogo. Quando um indivíduo interveniente num diálogo é plenamente visual mas sofre de incapacidade auditiva e/ou vocal, a necessidade de recorrer a gestos para se exprimir é imperativa. Esta problemática tem vindo a ser trabalhada ao longo dos anos, tendo sido desenvolvidas várias propostas para sistemas de reconhecimento de gestos que, de uma forma geral, devem ser capazes de captar, interpretar e categorizar a informação de estímulos adquiridos. Posteriormente, deverá ser capaz de executar estes três processos mediante novos estímulos, que, comparados com os existentes, devem resultar numa identificação. Este tipo de sistemas têm vindo a ser desenvolvidos e testados com diferentes sensores que permitem a captação e reconhecimento do movimento das mãos, do corpo, ou mesmo da expressão facial. Os dispositivos sensoriais de profundidade que recentemente surgiram no mercado, tais como, Microsoft Kinect [1], PrimeSense Carmine, ou estão a surgir, structure.io e Intel Real Sense, são bastante acessíveis no que respeita ao custo de aquisição. Estes dispositivo têm tido bastante relevância nas últimas abordagens ao problema de reconhecimento automático de línguas gestuais.

1.1. Motivação

A sociedade mundial está cada vez mais consciente das necessidades especiais de quem tem alguma deficiência física, existindo a intenção de várias entidades de criar sistemas e interfaces que optimizem a forma de acesso a meios informáticos por estes indivíduos. O reconhecimento e tradução de uma língua gestual é uma tarefa muito complexa mesmo para tradutores humanos com experiência. Devido à complexidade intrínseca das línguas, pela diversidade física dos executores ou pelas limitações tecnológicas, até aos dias de hoje, não é conhecida nenhuma solução óptima ou que resolva o problema na íntegra. As soluções apresentadas pela comunidade que recorrem a luvas, tais como,

data *glove*, *cyber glove*, *accele glove* [2, 3], entre outras, apresentam maior precisão na captação da forma e movimento da mão, mas apresentam grandes desvantagens, pois estas luvas são bastante dispendiosas, intrusivas e de fácil desgaste com a utilização. Outras soluções que recorrem a câmaras de vídeo, como por exemplo Webcams, para o reconhecimento da Língua Gestual Americana (*American Sign Language*, ASL) [4] ou da Língua Gestual Japonesa [5], revelaram-se insuficientes devido à baixa resolução e tempo de resposta. Contudo, novas abordagens que recorrem a dispositivos sensoriais de profundidade [6, 7, 8, 9] têm apresentado contribuições promissoras na área de reconhecimento automático de gestos aplicado a línguas gestuais, incluindo a Portuguesa (LGP) [10, 11, 12]. Actualmente é conhecida a intenção da Microsoft de desenvolver um sistema que traduza línguas gestuais com recurso ao Microsoft Kinect [13]. Durante a realização deste trabalho foi ainda disponibilizado no mercado o sistema UNI que realiza reconhecimento de ASL com recurso ao dispositivo Leap Motion [14].

No processo de comunicação, é necessário que o emissor e o receptor compreendam a mesma língua gestual. Tal como acontece nas línguas orais, não existe uma língua gestual universal e analogamente aos sotaques de uma língua oral de um país, numa língua gestual, é uma prática corrente a existência de pessoas do mesmo país mas de regiões distintas a utilizar gestos diferentes para as mesmas palavras. Desta forma, o conhecimento do código linguístico é essencial na comunidade escolar e fora dela. O estudo da língua gestual é um campo ainda em desenvolvimento mas existe o objectivo de melhorar e tornar mais eficaz o ensino, defendido pela UNICEF, UNESCO, PNUD e Banco Mundial, para “todas as pessoas – crianças, jovens e adultos” [15].

A utilização da tecnologia como forma de melhorar a comunicação e aquisição de competências da LGP, não só para quem sofre de incapacidade auditiva e/ou vocal mas também para indivíduos falantes, é a principal motivação para a realização deste trabalho científico.

1.2. Objectivos

Neste trabalho pretende-se desenvolver uma solução para colmatar as necessidades educativas especiais dos alunos surdos, para que estes possam complementar o ensino com uma ferramenta interactiva.

Pretende-se portanto, contribuir com uma solução que realize o reconhecimento e tradução automática de gestos em tempo real e de forma contínua aplicada à LGP. A solução não deve ter apenas em conta gestos estáticos (não requerem movimento) mas também gestos dinâmicos.

O primeiro desafio no desenvolvimento de um sistema de reconhecimento de gestos é a aquisição de estímulos naturais do corpo humano para que seja possível atribuir um significado a um gesto. Estes estímulos não são triviais de obter devido à necessidade de processar toda a informação adquirida em bruto e extrair apenas a que seja relevante para o contexto do problema. Após a aquisição e contextualização dos estímulos é necessário armazenar a informação para que seja possível posteriormente comparar e reconhecer novos gestos.

Pretende-se que o sistema adquira a informação de estímulos, das mãos e do corpo, com recurso a um sensor de profundidade, Microsoft Kinect. Posteriormente processe e modele essa informação de forma a extrair as características necessárias para a classificação. Por último, pretende-se testar o desempenho de algoritmos de reconhecimento de padrões e algoritmos baseados em aprendizagem automática no reconhecimento de gestos da LGP.

1.3. Contributos da tese

Neste trabalho são apresentados os seguintes contributos para o estado da arte:

- Construção de uma base de dados de gestos estáticos e dinâmicos da LGP realizados por um executor profissional;

- Utilização de duas técnicas clássicas de classificação de gestos estáticos aplicadas à LGP;
- Utilização de duas técnicas de classificação de gestos dinâmicos aplicadas à LGP;
- Desenvolvimento de uma aplicação que realiza o reconhecimento de gestos estáticos e dinâmicos em tempo real com segmentação automática.

1.4. Organização do documento

O restante conteúdo deste documento encontra-se organizado da seguinte forma:

Capítulo 2 - Neste capítulo são apresentadas as características gerais das línguas gestuais com ênfase na LGP e apresentados os aspectos estruturais mais relevantes a ela associados. É feito um resumo dos dispositivos e técnicas existentes para a aquisição de estímulos naturais do ser humano presentes no estado da arte e apresentados com maior detalhe os dispositivos sensorial Kinect e Leap Motion.

Capítulo 3 - No terceiro capítulo são apresentados alguns dos métodos mais relevantes descritos na literatura para o reconhecimento de gestos estáticos e dinâmicos e feita uma descrição mais detalhada dos métodos adoptados. São apresentados e classificados os diversos tipos de características possíveis de extrair do fluxo de imagens de vídeo, relatados na literatura como sendo utilizados para a caracterização de um gesto. Por último, é apresentada uma síntese dos métodos e técnicas de reconhecimento de gestos presentes na literatura. São apresentadas soluções que utilizam diferentes dispositivos para a captura de informação e apresentadas propostas que utilizam métodos com maior aplicabilidade para a extracção e reconhecimento de gestos estáticos e dinâmicos.

Capítulo 4 - Neste capítulo é apresentada a arquitectura adoptada para o sistema proposto e feita uma descrição geral dos seus componentes. É feito um resumo dos métodos experimentados para o acesso aos dipositivos sensoriais e apresentada a razão da escolha do sensor a utilizar. De seguida, são apresentados os métodos de

classificação adoptados, descritas as ferramentas utilizadas no desenvolvimento e apresentada a interface de utilização.

Capítulo 5 - No quinto capítulo é introduzida e apresentada em detalhe, a abordagem adoptada para o desenvolvimento do sistema proposto. Inicialmente é apresentada a forma de acesso ao dispositivo sensorial e de seguida são apresentadas as características da mão e do corpo utilizadas e de que forma são extraídas. Ainda neste capítulo é descrita a implementação dos algoritmos de classificação de gestos estáticos e dinâmicos e apresentada a forma de interacção com cada um deles. De seguida é apresentado o método de reconhecimento contínuo adoptado e por último é apresentado o processo da criação da base de dados de gestos estáticos e dinâmicos da Língua Gestual Portuguesa.

Capítulo 6 - Neste capítulo são apresentados os resultados obtidos através dos testes aos algoritmos, apresentados no capítulo anterior, de classificação de gestos estáticos e dinâmicos.

Capítulo 7 - Este capítulo contém as conclusões, considerações e resultados obtidos. Por último, são apresentadas algumas recomendações de trabalho futuro.

2. Fundamentos do trabalho

Neste capítulo são apresentados conceitos fundamentais apreendidos na literatura existente para a compreensão deste trabalho.

2.1. A Língua Gestual

As línguas gestuais ou, pelo menos, os sistemas de comunicação gestual, terão surgido em diversos contextos e entre grupos de surdos. Já na idade média, aos monges em clausura em mosteiros europeus, tendo feito votos de silêncio, apenas lhes era permitida a comunicação entre si por via da língua gestual [16]. Por força da necessidade, ao longo dos anos, as comunidades de surdos de todo o mundo foram desenvolvendo as suas próprias línguas gestuais com estruturas gramaticais e gestos muito próprios. É portanto uma prática corrente, a existência de pessoas de regiões distintas, a utilizar gestos diferentes para as mesmas palavras. Por estas razões, a língua gestual tornou-se uma marca importante da identidade das pessoas surdas e constitui um elemento mais unificador na comunidade, enquanto meio de transmissão de valores e da herança cultural dessa comunidade [17].

2.2. Língua Gestual Portuguesa

A necessidade de remover as barreiras que impedem o reconhecimento das línguas gestuais tem estado presente em diversos documentos de cariz internacional. Os Estados Unidos da América, a Holanda e a Suécia têm contribuído desde 1960 para que as línguas gestuais sejam progressivamente reconhecidas em vários países, renovando a concepção da linguagem, desde sempre atribuída à correspondência de som ao significado. Em Portugal esta problemática tem vindo a ser trabalhada cientificamente desde 1994 [18] e em 1997 a LGP passou a ser uma das línguas oficiais de Portugal segundo a Constituição Portuguesa, juntamente com a Língua Portuguesa e o Mirandês. A LGP é portanto uma língua recente, sendo representada através do gesto,

percepcionada pelo sentido da visão e realizada num espaço tridimensional, onde se marca a dinâmica da localização e orientação da mão, dos movimentos, configurações, expressão facial e corporal [19].

2.2.1. Estrutura da Língua Gestual Portuguesa

Tal como qualquer outra língua gestual, a LGP é expressa por gestos, movimento das mãos, do rosto, do corpo e da expressão facial, mas os seus componentes mínimos são essencialmente elementos manuais [20], isto é, produção motora da mão e do corpo [21]. Na LGP existem três tipos de gestos, os *icónicos*, os *referenciais* e os *arbitrários.* Os primeiros são representados através do delinear do objecto (desenhar uma forma) ou pela configuração das mãos para representar o próprio objecto atribuindo-lhe dimensão e forma ou para representar uma acção sobre o objecto. Os gestos referenciais são utilizados para representar todos os pronomes pessoais, partes do vestuário ou do corpo, e por último, os gestos arbitrários incluem todos aqueles que não apresentam qualquer relação directa com o conceito que representa [18].

Os gestos, na sua natureza manual, são representados através das diferentes configurações de mão, na sua orientação (palma virada para cima ou para baixo), na localização da mão em relação ao corpo (no tronco, nos membros superiores, na cabeça e espaço neutro, com ou sem contacto), no tipo de movimento das mãos e braços (rectilíneo, circular, para cima e para baixo, etc). A estas características é ainda possível considerar a relação entre as mãos, uma vez que, existem gestos que são representados apenas com uma das mãos e outros com ambas. É portanto possível considerar se existe contacto entre as mãos, se uma ou ambas estão em contacto com o corpo ou se as duas mãos têm igual configuração e movimentos iguais ou simétricos [18, 20].

Figura 1 - Alfabeto manual da Língua gestual Portuguesa (retirado: APS)

Em relação a componentes não manuais, a expressão facial é o canal privilegiado para expressar sentimentos e emoções, permite distinguir lexicalmente os gestos ou formular frases interrogativas, negativas e condicionais. Nos casos em que, gestos que são executados da mesma forma mas têm significado diferente, a expressão facial funciona como elemento distintivo. Por último, a direcção do olhar pode representar a localização, atribuir o significado de discurso entre dois ou mais intervenientes, dar ênfase ao diálogo ou chamar a atenção.

2.3. Dispositivos de reconhecimento

O paradigma da interacção pessoa-máquina (*human-computer interaction,* HCI), tem vindo a ser aperfeiçoado mas não se alterou significativamente, em termos de utilização generalizada. Ainda hoje, o teclado e o rato são a forma mais disseminada de *input* na HCI, estando presente em qualquer computador. O reconhecimento de elementos

naturais do ser humano, isto é, reconhecimento de gestos, da expressão facial, de voz e do movimento dos olhos são componentes que a comunidade se refere como *perceptual user interfaces* (PUIs). Esta área foca-se em combinar capacidades humanas naturais de comunicação, motora, cognitiva e a capacidade de as utilizar com o *I/O* de um computador, percepção automática e processamento [22].

Relativamente ao reconhecimento de uma língua gestual, uma vez que, os seus componentes mínimos são essencialmente elementos manuais, os diversos trabalhos desenvolvidos ao longo de anos focam-se essencialmente em percepcionar a mão humana com recurso a diversos dispositivos e sensores. Existem dois grandes grupos que classificam estes tipos de dispositivos e sensores, os baseados em vídeo, que permitem ao executor mover-se livremente sem ter nada agarrado ao corpo, e os intrusivos, que requerem sempre algum dispositivo físico em contacto com o corpo.

Relativamente aos dispositivos intrusivos, as abordagens baseadas em luvas para a percepção dos estímulos da mão, *Data Entry Glove*, *CyberGlove* e *AcceleGlove* [2, 3, 23], obtiveram maior sucesso na precisão do que as abordagens que utilizam dispositivos baseados em vídeo. Isto porque, combinadas com dispositivos infravermelhos, sensores ultra-sónicos ou magnéticos, permitem obter uma precisão de centímetros (ultra-sónico) ou milímetros (magnético). O grande problema destas luvas, para além de serem muito dispendiosas e intrusivas, é a necessidade do executor permanecer no raio de alcance e num ambiente controlado, livre de interferências (magnéticas ou luminosas) ou interrupções do campo de visão.

No que diz respeito a obter informação da mão, os dispositivos baseados em vídeo têm tido bastante foco pois permitem ao executor mover-se livremente sem ter nenhum dispositivo em contacto com o corpo. A trajectória, forma da mão e a sua localização são detectadas por uma câmara de vídeo 2D ou 3D, existindo também a necessidade do executor permanecer perto e num ambiente também relativamente controlado. A necessidade de processar grandes quantidades de informação para extrair e seguir as mãos na imagem introduz uma restrição na memória, velocidade de processamento e complexidade [7, 24]. Para o reconhecimento de gestos, os dispositivos sensoriais de profundidade conseguem calcular a distância para um dado objecto disponibilizando um

modelo a três dimensões do seu campo de visão, sendo bastante promissores e alvo de vários estudos [8, 25, 26, 27].

2.3.1. Dispositivo Kinect

A Microsoft ao adquirir a tecnologia sensorial 3D à empresa Israelita PrimeSense conseguiu em Novembro de 2010 introduzir no mercado o sensor Kinect (Figura 2) como sendo um dispositivo de *input* para a consola de jogos Xbox 360. Até Fevereiro de 2011 contou mais de 24 milhões de dispositivos vendidos [28].

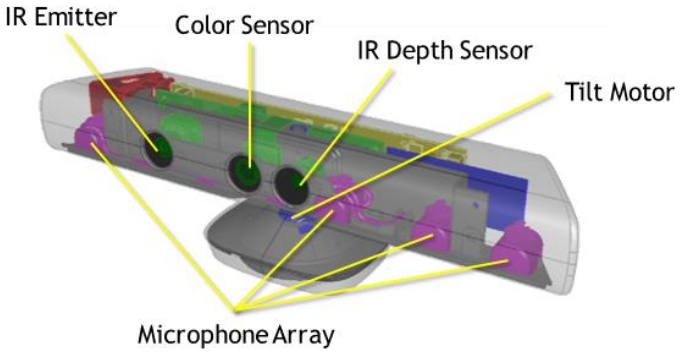


Figura 2 - Dispositivo Kinect

Este sensor permite captar movimentos do corpo humano a três dimensões e reconhecimento de voz, enquadrando-se no paradigma da *natural user interface* (NUI). As NUIs focam-se em desenvolver HCI baseados nas capacidades naturais do ser humano.

2.3.1.1. Tecnologia do sensor Kinect

O princípio base por detrás do sensor de profundidade Kinect é a emissão de um padrão contínuo de infravermelhos e simultaneamente capturar a imagem IR com uma câmara CMOS equipada com um filtro IR. O processador de imagem do Kinect utiliza a posição

relativa dos pontos no padrão para calcular o deslocamento de profundidade para cada *pixel* da imagem [25] fornecendo coordenadas (x, y, z) de objectos em três dimensões. O sistema de coordenadas do dispositivo é apresentado na Figura 3.

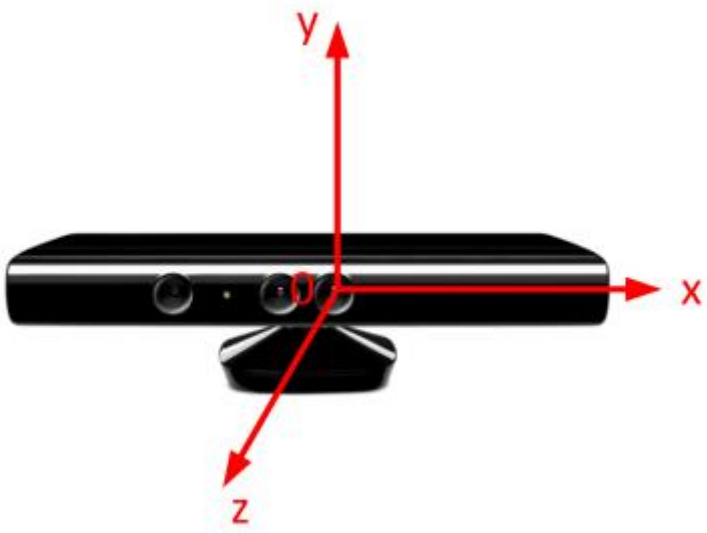


Figura 3- Sistema de coordenadas Kinect

O seu campo de visão é de 57° na horizontal, 43° na vertical. Para além do fluxo de imagens de vídeo de profundidade de 11-bit (depois de processamento) com a resolução de 640x480 a 30 imagens por segundo (*frames per second*, FPS), este sensor disponibiliza um fluxo de imagens VGA de 8 bit de profundidade de cor, proveniente da câmara RGB, com a resolução de 640x480 a 30 FPS. Os *drivers* podem ainda obter fluxos de imagens de 1280x1024 mas no entanto ficam limitadas a 10 FPS. O campo de visão para a sua utilização é de aproximadamente de 0,8m a 3,5m e a resolução de profundidade é cerca de 1,3mm por *pixel* para distâncias curtas (0,8m a 1,5m) e de 1cm para distâncias de 2m a 3,5m. O Kinect vem também equipado com quatro microfones e disponibiliza quatro canais de áudio com 32-bit a uma taxa de amostragem de 16KHz, permite localização acústica e supressão de ruído. O Kinect é ainda capaz de detectar até seis pessoas, mas apenas permite dois utilizadores activos para análise, conseguindo extrair a informação de vinte articulações de cada um deles [29]. A organização destes componentes no dispositivo é apresentada na Figura 2.

A primeira iniciativa de utilizar o sensor Kinect num computador, nasceu em 2010 numa "comunidade de *hackers*" que realizou *reverse-engeneering* à cadeia de dados provenientes da interface USB. Assim surgiu a comunidade OpenKinect [30], que se dedica a desenvolver e a disponibilizar *software* multiplataforma, gratuito e de código aberto. Em Novembro 2010, a empresa PrimeSense, que inicialmente desenvolveu a tecnologia do Kinect, aliou-se ao consórcio, sem fins lucrativos, OpenNI [31] e disponibilizaram o seu primeiro *software development kit* (SDK). Este SDK é multiplataforma e foi desenvolvido para ser independente do sensor utilizado, podendo ser utilizado não só com o Kinect mas também com os sensores da PrimeSense ou ASUS. Finalmente, em Junho de 2011 a Microsoft lançou o seu SDK permitindo que este dispositivo seja utilizado como ferramenta para produtos não comerciais [1]. Este SDK apenas foi disponibilizado para Windows 7 ou superior e apenas fornece valores de profundidade até 4m, enquanto as outras duas *frameworks* fornecem valores a mais de 9m. Outra diferença é que o SDK da Microsoft está limitado para profundidades inferiores a 0,8m enquanto as outras duas *frameworks* são capazes de reconhecer desde os 0,5m.

2.3.2. Dispositivo Leap Motion

A Leap Motion, Inc. é a empresa que desenvolveu e comercializa, desde Julho de 2013, o dispositivo sensorial de custo reduzido Leap Motion, apresentado na Figura 4. Este sensor comunica com um computador através de uma interface USB, adquire e reconhece o movimento das mãos e dos dedos, sem necessidade de contacto físico, fornecendo um modelo a três dimensões do que percepciona. Este dispositivo está equipado com duas câmaras de IR monocromáticas, três LEDs IR e o seu campo de visão é de sensivelmente de 25mm a 600mm numa área hemisférica. Os três LEDs geram um padrão de pontos a três dimensões que por sua vez são captados pelas câmaras gerando quase 300 imagens por segundo à entrada da interface USB. Por fim, depois de adquiridas as imagens, estas são analisadas e processadas pelo controlador (*software*), proprietário da Leap Motion, para o reconhecimento das mãos e dos dedos. O *software* proprietário da Leap Motion disponibiliza um serviço (Windows) ou um *daemon* (Mac

ou Linux) que permite a outras aplicações acederem à informação das mãos e dos dedos após processamento (*tracking data*). O SDK da Leap Motion foi desenvolvido em duas variantes da API para acesso à *tracking data*. Na primeira foram especificadas interfaces nativas, disponibilizadas como bibliotecas que podem ser incluídas em aplicações que pretendam obter a informação disponibilizada pelo serviço ou *daemon*. Na segunda variante foram definidas interfaces *WebSocket* que, por exemplo, se for utilizada em conjunto com a biblioteca cliente JavaScript, permite criar aplicações *Web* que tirem partido deste dispositivo.

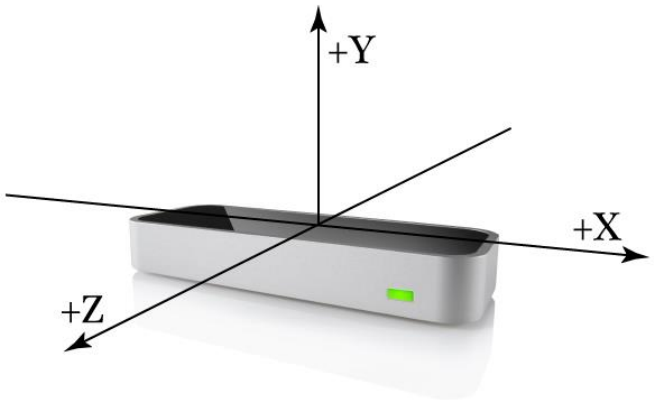


Figura 4 - Sistema de coordenadas Leap Motion

A Leap Motion Inc., até aos dias de hoje ainda não divulgou mais detalhes sobre o processamento que o controlador realiza. Este sensor diferencia-se do Kinect pelo curto campo de visão e a alta resolução permitindo que a utilização mais adequada seja na realização de tarefas como navegar numa página de Internet, fazer *zoom* num mapa ou produzir desenho de precisão [32].

3. Estado da Arte

Neste capítulo é apresentada uma síntese de propostas retiradas da literatura existente no que diz respeito a dispositivos e métodos utilizados no reconhecimento de gestos estáticos e dinâmicos. Inicialmente são descritas propostas que tiram partido de diferentes dispositivos para a aquisição da informação. De seguida são apresentados alguns métodos e técnicas propostas para o reconhecimento de gestos estáticos e dinâmicos de uma língua gestual.

3.1. Métodos de reconhecimento

No reconhecimento de gestos, têm sido utilizados vários métodos, dos quais se destacam, *template matching* [9, 10, 11], correspondência estatística [33], correspondência linguística, análise de componentes principais (*principal component analisys*, PCA) [34, 35, 36], alinhamento temporal dinâmico (*dynamic time warping*, DTW) [37], redes neuronais [3, 36, 38], modelos de Markov não-observáveis (*hidden Markov models*, HMM) [3, 6, 39] e métodos *ad hoc*. O método de *template matching* apresenta a vantagem de ser fácil de treinar porque os protótipos das classes ou gestos que se pretendem reconhecer são simplesmente padrões de exemplo. No entanto, a necessidade de utilizar um grande número de protótipos para tornar o sistema mais flexível às várias formas de apresentação dos gestos, nomeadamente, em relação a rotações e escalamentos, pode levar a que este método seja considerado computacionalmente inapropriado.

Nos métodos que utilizam correspondência estatística são construídos classificadores a partir da frequência de ocorrência nos vectores de características do conjunto de treino. As estatísticas mais utilizadas são as médias dos vectores de características das várias classes, as variâncias de características individuais por classe e a correlação de características dentro de uma classe. Em alguns métodos de correspondência estatística assume-se conhecimento *a priori* sobre a distribuição de características de uma classe. No entanto, quando estas hipóteses não são cumpridas, a performance do sistema é

penalizada. Nos métodos em que não existe informação *a priori* sobre a distribuição, é necessário mais informação de treino para estimar a sua forma.

A aproximação linguística tenta aplicar autómatos e a teoria da língua formal no problema de reconhecimento de modelos. O grande problema desta abordagem é a necessidade de ser fornecida uma gramática para cada classe modelo.

O método de análise de componentes principais é uma técnica para encontrar padrões na informação e expressá-la de forma a destacar a suas semelhanças e diferenças. A grande vantagem do PCA é que assim que são encontrados os padrões é possível comprimir a informação, reduzindo o número de dimensões sem grande perda de informação. Após este processo, é possível utilizar essa informação como características a fornecer a algoritmos de classificação.

O DTW é um método que calcula a melhor correspondência entre duas sequências temporais, mesmo que estas variem na duração ou na velocidade. Neste método, as sequências são comprimidas ou expandidas ("*warped*") não linearmente no tempo para que seja possível determinar uma medida de semelhança, independentemente da velocidade com que as sequências foram geradas.

As redes neuronais têm sido aplicadas com sucesso para resolver vários problemas de reconhecimento de modelos e a sua principal vantagem é a de ser construída a partir de um grande número de elementos simples capazes de aprender e de resolver problemas complicados e ambíguos, colectivamente. A grande desvantagem desta técnica é a tendência para a necessidade de utilizar uma grande capacidade de processamento, especialmente para o treino.

Um dos métodos mais populares de aprendizagem automática em reconhecimento de gestos corresponde a representar o gesto como uma sequência de observações de um HMM, sendo um dos métodos que recebeu maior atenção da literatura de classificação de gestos [4, 6, 11, 39, 40, 41]. Um HMM é uma ferramenta estatística para modelar sequências generativas que podem ser caracterizadas por um processo subjacente de geração de sequências observáveis [42]. Esta técnica foi utilizada com sucesso em várias aplicações de reconhecimento de fala. Os movimentos do corpo, tal como os sons

produzidos pelo trato vocal podem ter uma associação à produção de gestos, introduzindo regras sintácticas semelhantes à fala. Como um gesto é um movimento expressivo, é natural descrevê-lo como um movimento através de um modelo sequencial. Os HMMs são uma escolha atractiva para processar informação tridimensional de gestos porque a sua natureza baseada em estados permite descrever como um gesto evolui ao longo do tempo. Permite também modelar variações na duração dos gestos através da permanência ou comutação entre estados.

3.1.1. Template Matching

O *template matching* é uma das técnicas que nós, humanos, utilizamos excepcionalmente bem e por ser tão fundamental existem inúmeros trabalhos que apresentam abordagens para prover uma máquina de tal capacidade. A aplicação desta técnica é simples, onde para classificar um novo padrão P', apenas é necessário correlacioná-lo com os diversos padrões de exemplo, presentes num conjunto de treino $P = \{P_1, P_2, \dots, P_{n\}}$, de forma a determinar um grau de semelhança entre eles. No método de reconhecimento mais simples, para cada padrão de exemplo é produzida uma classe C não podendo existir classes vazias, $C \neq \emptyset$. De forma a aumentar a probabilidade de semelhança no reconhecimento, cada classe pode ser representada como um subconjunto de P sendo $C' \subseteq P$, ou seja, cada classe pode conter n padrões de exemplo. Para classificar P'é necessário compará-lo com cada elemento de P sendo-lhe atribuída a classe do elemento que apresentar maior semelhança. A eficiência computacional deste método, para o reconhecimento de gestos, tem vindo a ser questionada durante várias décadas. Isto porque, uma vez que o tempo necessário para comparar P' com todo o conjunto P tende a aumentar proporcionalmente com a cardinalidade de P pois o custo desta operação é $O(n)$. Existem inúmeras abordagens que tentam diminuir a cardinalidade de P e por sua vez diminuir o tempo de processamento despendido na classificação. De uma forma geral, todas elas tentam normalizar o conjunto de exemplos ou aplicar operações que compensem variações irrelevantes na posição, tamanho ou orientação. A produção de um padrão médio $\bar{P}$ por classe é também apontado como uma boa opção para diminuir o número de

comparações na pesquisa, funcionando $\bar{P}$ como um filtro de classes. Outra forma de melhorar o tempo despendido na classificação é tirar partido de *hardware* com *multi-core* de forma a paralelizar a operação de correlação de padrões.

3.1.2. Análise de Componentes Principais

O método de análise de componentes principais é uma técnica para encontrar padrões na informação e expressá-la de forma a destacar a suas semelhanças e diferenças. Esta técnica, até aos dias de hoje, foi utilizada com sucesso em várias áreas como, processamento de imagem, análise de informação ou pré-processamento de informação. Encontrar padrões pode ser uma tarefa difícil quando a informação contém muitas dimensões ou quando não está disponível uma representação gráfica. Nestes casos o PCA é uma ferramenta adequada para a análise.

A projecção da informação nos primeiros vectores ortogonais obtidos pela decomposição própria da matriz de covariância, apresentada em (1), é a forma de alcançar o objectivo anteriormente descrito.

$$C = \begin{pmatrix} cov(x,x) & cov(x,y) & cov(x,z) \\ cov(y,x) & cov(y,y) & cov(y,z) \\ cov(z,x) & cov(z,y) & cov(z,z) \end{pmatrix} \quad (1)$$

Em primeiro lugar é necessário representar a informação como um vector de N dimensões. De seguida, é necessário normalizar a informação calculando e subtraindo a média $\bar{X}$ de cada dimensão X. Após normalizados os vectores, é necessário medir a relação entre cada duas dimensões. Desta forma, obtém-se uma medida representativa de como cada dimensão varia em relação a outra. Se o valor obtido for positivo então significa que as duas dimensões aumentam simultaneamente. No caso em que o valor obtido é negativo significa que quando uma dimensão aumenta a outra diminui. No caso em que a covariância é zero, indica que as duas dimensões são independentes uma da

outra. Uma vez que a covariância é medida entre duas dimensões, se tivermos três dimensões (x, y, z), então é necessário medir a covariância entre as dimensões x e y, x e z e por último y e z. Esta informação é representada pela matriz de covariância que pode ser estimada pela seguinte equação:

$$cov(X,Y) = \frac{\sum_{i=1}^{n}(Xi - \bar{X})(Yi - \bar{Y})}{(n-1)} \quad (2)$$

Uma forma de representar a co-variância em dados com várias dimensões é através de uma matriz C, denominada matriz de covariância (equação 1).

Pode-se representar um determinado vector de características numa combinação linear em que a base é constituída pelas componentes principais. Estas componentes principais são os m vectores próprios da matriz de covariância, associada aos m maiores valores próprios, onde m está limitado pelo número de padrões de treino. Para uma matriz quadrada A com dimensão m x m, um vector v de comprimento m diz-se que é vector próprio de A se e só se satisfizer a equação $Av = \lambda v$, onde λ é um número real que quando multiplicado por v resulta no mesmo vector que Av. Quando se obtém um vector próprio v para a matriz A, o valor λ associado é denominado de valor próprio do vector próprio v. Por último para classificar um novo vector de informação basta simplesmente projectar o vector no espaço de componentes principais através da equação (3).

$$y = W^T \times (X - \bar{X}) \quad (3)$$

onde W corresponde à matriz dos vectores próprios, e calcular a distância vectorial entre o resultado da projecção e cada um dos vectores principais. A menor distância encontrada representa a maior semelhança entre os vectores [34].

3.1.3. Alinhamento Temporal Dinâmico

O método de alinhamento temporal dinâmico ganhou popularidade por ser extremamente eficiente em medir semelhanças em séries temporais, minimizando os efeitos de deslocamento e distorção temporal. Este algoritmo é utilizado em diferentes áreas, tais como, reconhecimento de fala, de escrita, de gestos, entre outros. Dadas duas sequências temporais $X = \{x_1, x_2, \dots, x_N\}$ e $Y = \{y_1, y_2, \dots, y_M\}$, vector de referência e vector de teste respectivamente, com $M\ e\ N\ \in\ \mathbb{N}$, o objectivo é alinhar as duas sequências no tempo através de mapeamento não-linear (*warping*). Este alinhamento é realizado com recurso a uma função de distância ou custo, sendo o custo total C_p o somatório de todas as distâncias entre os elementos mapeados. Assim, o algoritmo começa por construir uma matriz $C\ \in\ \mathbb{R}^{N\times M}$, denominada de matriz de custo local, contendo todos os pares de distâncias entre X e Y para que seja possível alinhar as duas sequências.

$$C_l\ \in\ \mathbb{R}^{N\times M}:\ c_{i,j} =\ \|x_i, y_i\|, i\ \in [1:N], j\ \in [1:M] \tag{4}$$

Após construída a matriz de custo local, o algoritmo encontra o caminho de alinhamento (*warping path*) e define a correspondência de um elemento $x_i \in X$ ao $y_j \in Y$ seguindo a condição de limite. Este caminho é o que atravessa as áreas de menor custo na matriz C e a condição de limite define quais os primeiros e últimos elementos de X e de Y. A função de custo associada ao caminho de alinhamento é dada pela soma de todos os custos entre os elementos da sequência.

$$C_p(X, Y) = \sum_{l=1}^{L} c(x_{nl}, y_{ml}) \tag{5}$$

O caminho que tiver menor custo associado ao alinhamento é denominado de caminho óptimo de deformação P^* (*optimal warping path*). Seguindo a definição de caminho

óptimo de deformação, para encontrá-lo é necessário testar todos os caminhos de deformação possíveis entre X e Y. Isto pode ser computacionalmente dispendioso devido ao crescimentos exponencial do número de caminhos óptimos quando os comprimentos de X e Y crescem linearmente. Para colmatar este problema, o DTW aplica um algoritmo baseado em programação dinâmica que utiliza a função de distância do DTW com um custo computacional de apenas $O(MN)$.

$$DTW(X,Y) = c_{p^*}(X,Y) = \min\{c_p(X,Y), p \in P^{N\times M}\} \quad (6)$$

onde $P^{N\times M}$ é o conjunto de todos os possíveis caminhos de deformação sendo criada uma matriz de custo acumulado D que é definida como:

1. Primeira linha: $D(1,j) = \sum_{k=1}^{j} c(x_1, y_k), j \in [1,M]$
2. Primeira coluna: $D(i,1) = \sum_{k=1}^{i} c(x_k, y_1), i \in [1,N]$
3. Todos os outros elementos: $D(i,j) = \min\{D(i-1,j-1), D(i-1,j), D(i, j-1)\} + c(x_i, y_j), i \in [1,N], j \in [1,M]$

Depois de construída a matriz D, o caminho de deformação pode ser obtido simplesmente retrocedendo a partir do ponto $p_{end} = (M, N)$ até ao ponto inicial $p_{start} = (1, 1)$ seguindo a direcção do índice que apresentar menor custo [43].

3.1.4. Modelo de Markov Não-Observável

Os modelos de Markov não-observáveis são modelos probabilísticos adequados para modelar comportamentos temporais, assumindo que esses comportamentos contêm um número finito de estados internos. São denominados de escondidos porque não se observam directamente os estados mas uma sequência de observações por eles gerada. Desta forma, pretende-se descobrir uma sequência adequada de estados para suportar as observações. Um HMM é representado por uma colecção finita de estados ligados

por transições, onde cada estado é caracterizado por dois conjuntos de probabilidades. O primeiro é a probabilidade de transição entre estados e o segundo é a distribuição de probabilidade discreta de saída ou densidade de probabilidade de saída, associada à geração das observações, discretas ou contínuas, respectivamente. O modelo define, para um dado estado, qual a probabilidade de produzir na saída (observação) cada um dos símbolos presente no alfabeto finito ou um vector contínuo aleatório [41, 42].

A definição formal de um HMM é:

$$\lambda = (A, B, \pi) \tag{7}$$

onde A define a matriz de transição entre estados, B define a matriz de probabilidade de observações (distribuição de probabilidade ou função densidade probabilidade) e π, define o vector de probabilidades do estado inicial. Associados a estas matrizes existem os conjuntos S e V, onde S é conjunto de estados e V é o conjunto de funções densidade de probabilidade:

$$S = (s_1, s_2, \dots, s_n) \tag{8}$$

$$V = (v_1, v_2, \dots, v_m) \tag{9}$$

É definido Q como uma sequência fixa de estados de comprimento T, e O como as observações correspondentes:

$$Q = q_1, q_2, \dots, q_T \tag{10}$$

$$O = o_1, o_2, \dots, o_T \tag{11}$$

O sistema está num dos estados do HMM num determinado instante temporal e transita para um novo estado regularmente em intervalos espaçados de tempo. Cada transição de estado S_i para S_j tem associada uma determinada probabilidade a_{ij} sendo $\sum_i a_{ij} = 1$. A matriz de transições A armazena a probabilidade de um S_j ser sucessor do estado S_i. As transições de estado são independentes temporalmente:

$$A = [a_{ij}], a_{ij} = P\big(q_t = s_j \big| q_{t-1} = s_i\big) \quad (12)$$

B é a matriz de observações que armazena a probabilidade de observação k ser produzida a partir do estado S_j independentemente de t:

$$B = [b_i(k)], b_i(k) = P(x_t = v_k | q_t = s_i) \quad (13)$$

π é o vector de probabilidade inicial:

$$\pi = [\pi_i], \pi_i = P(q_1 = s_i) \quad (14)$$

No HMM é assumido que um estado é apenas dependente do estado anterior, chamada de propriedade de Markov, que representa a memória do modelo. É ainda assumido que a observação de saída num instante temporal o_t é apenas dependente do estado actual, sendo independente das observações e estados anteriores.

Os HMMs de uma forma geral são utilizados para resolver três problemas, o da avaliação, o da descodificação e o da aprendizagem. O processo de aprendizagem destina-se a estabelecer modelos de gestos de acordo com a informação de treino através do algoritmo de Baum-Welch [42]. Dado um conjunto de exemplos de um processo, este algoritmo permite estimar os parâmetros do modelo $\lambda = (A, B, \pi)$ que

melhor descrevem o processo. Existem duas aproximações padrão para esta tarefa, o treino supervisionado e o não supervisionado. Se os exemplos de treino contiverem as observações e as respectivas classes, pode ser aplicado o treino supervisionado, igualando as observações às entradas e as classes às saídas. No caso em que apenas as entradas são fornecidas como informação de treino, deve ser utilizado treino não supervisionado para determinar o modelo que produz as observações com maior probabilidade. No processo de avaliação é possível atribuir um valor à correspondência entre um modelo λ e uma sequência de observações O, ou seja, dado um HMM e uma sequência de observações é necessário calcular a probabilidade dessa sequência ser gerada pelo modelo, ou seja, $P(O|\lambda)$. Este problema pode ser visto como o de avaliar o quão correcto um modelo produz uma dada sequência de observações, permitindo-nos escolher o modelo mais apropriado de um conjunto. No contexto do reconhecimento de gestos, esta avaliação pode ser utilizada para o reconhecimento de gestos isolados. O processo de descodificação permite encontrar a sequência de estados que produziu uma dada sequência de observações com maior probabilidade. Uma solução para este problema é a utilização do algoritmo de Viterbi [42] que permite encontrar a sequência mais provável de estados para uma sequência de observações. Esta técnica pode ser utilizada para o reconhecimento de gestos contínuos determinando o modelo ou classe que gera as respectivas sequências de observações com maior probabilidade.

Uma limitação do modelo descrito é que as observações assumem ser características unidimensionais. No entanto no problema do reconhecimento de gestos algumas características são obtidas e modeladas por vectores de duas ou mais dimensões. Uma solução para este problema é utilizar um modelo multidimensional que assume a independência de características observadas.

Um gesto executado por um caminho pode facilmente ser decomposto em séries temporais 2D ou 3D no espaço Cartesiano onde um caminho do gesto $g(x, y, z, t)$ pode ser decomposto em $X(t)$, $Y(t)$ e Z(t). Para além disto, um HMM multidimensional permite ainda a utilização de múltiplas características de forma a aumentar a taxa de reconhecimento [41].

3.2. Extracção de características

A extracção das características do fluxo de imagens de vídeo é crucial porque os gestos da mão são muito ricos na variação de forma, movimento e texturas. Os gestos podem ser representados através de várias características, tais como, padrões, transformações globais, localização onde é executado (zonas) ou através de características geométricas [41]. Os padrões são as características mais simples de calcular uma vez que são partes ou o todo da informação adquirida em bruto. As transformações globais, tais como, a rotação, translação ou escalamentos podem ser aplicadas de forma a reduzir o número de características dos padrões. As zonas são uma forma de derivar características de um caminho (sequência) pelo qual um gesto foi executado. O espaço é dividido num número limitado de zonas e o caminho percorrido pela mão é transformado em sequências de zonas. As características geométricas de um caminho, tal como, o comprimento total do gesto, o total dos ângulos formados pelo caminho ou o número de vezes que o caminho cruza com ele próprio podem ser utilizados para representar as propriedades globais do caminho. Para representar a postura da mão, usualmente são extraídas características de baixo nível que incluem o centróide da mão, a região delimitadora desta, a localização das pontas dos dedos e a sua direcção [5, 10, 11]. Estas últimas, em alguns casos não estão disponíveis ou não são extraídas com um grau de confiança suficientemente elevado devido a não estarem visíveis ou não serem percepcionadas por condições luminosas apropriadas. Existem outras características não geométricas para representar a mão, como por exemplo, a cor, a silhueta ou a textura, contudo revelam-se pouco adequadas para um reconhecimento robusto.

3.3. Sensores e dispositivos

Relativamente ao reconhecimento de uma língua gestual, uma vez que, os seus componentes mínimos são essencialmente elementos manuais, os diversos trabalhos desenvolvidos ao longo de anos focam-se essencialmente em percepcionar a mão humana com recurso a diversos dispositivos e sensores. Existem dois grandes grupos que classificam estes tipos de dispositivos e sensores, os baseados em vídeo, que

permitem ao executor mover-se livremente sem ter nada agarrado ao corpo, e os intrusivos, que requerem sempre algum dispositivo físico em contacto com o corpo.

Hernández-Rebollar et al. em [2], apresentam um método capaz de reconhecer 26 gestos estáticos da ASL com base na informação dos dedos disponibilizada por uma *Accele Glove*. A extracção e modelação das características de um gesto segue a definição proposta por Stokoe [44] e Costello [45] onde um gesto pode ser representado pela forma da mão e pela sua orientação. Para a classificação utilizam um algoritmo de três níveis, onde no primeiro, a orientação da mão é discriminada em três subclasses, vertical, horizontal ou fechada. No segundo nível cada subclasse anterior é novamente discriminada mediante as coordenadas a duas dimensões das pontas dos dedos. Desta forma é possível excluir da comparação um grande conjunto de gestos que não coincide com esta discriminação. Por último, caso a discriminação da segunda camada não seja suficiente para reconhecer o gesto é então utilizada uma terceira camada. Nesta última são pesquisadas diferenças particulares na informação em bruto para obter a classificação final. Este sistema apresentou uma precisão de 100% para 21 das letras e o pior cenário é referente à letra 'U' com 78% de reconhecimentos correctos. Neste sistema foram adicionados dois símbolos, 'espaço' e 'introduzir' permitindo ao utilizador formular frases.

No trabalho de Ed Kaiser et al. [24], é apresentada uma arquitectura de interacção 3D multimodal para redução de erros de classificação utilizando informação estatística derivada da fala, dos gestos e do meio ambiente. Para a classificação de gestos, é necessário que o executor coloque quatro sensores no corpo que fornecem informação do braço e da cabeça. Depois de analisada a informação é passada ao agente de reconhecimento de gestos 3D que com base em regras pré-definidas consegue distinguir 4 gestos com uma precisão de 75,7%.

No trabalho de Liviu Vladutu [35], é apresentado um novo método para identificar gestos de uma Língua Gestual recorrendo a uma câmara de vídeo a cores. Este novo método pode ser descrito em 3 módulos. No primeiro módulo são calculadas as componentes principais das imagens de treino, aplicada decomposição de valores singulares que por sua vez são utilizados para seleccionar as imagens mais

representativas da sequência de observações através de lógica difusa. No segundo módulo são extraídas características a partir de descritores de modelos de cor e forma de regiões desenvolvido pelo grupo MPEG-7. Por último, para obter uma classificação de um novo gesto é realizada uma modelação não linear dos dados recorrendo a máquinas de suporte vectorial.

Thad Starner et al. em [4], apresentam uma abordagem que recorre a um chapéu de basebol com uma câmara a cores embutida para reconhecer ASL ao nível da frase. Este sistema apesar de ser baseado em vídeo pode também ser considerado intrusivo, uma vez que, a câmara está embutida num chapéu que o executor terá que colocar na cabeça. Para a extracção das características da mão foram testadas duas hipóteses, a de utilizar uma luva ou a de detectar a mão com base na cor da pele. Para a classificação, utilizam HMMs que recebem como características normalizadas, a descrição da forma da mão, a sua orientação e trajectória. O sistema apresentou uma precisão superior a 97% por palavra para um léxico de 40 palavras.

Yi Li em [7] apresenta uma proposta para um sistema de reconhecimento que utiliza o dispositivo Microsoft Kinect para a aquisição de informação. Com base em cada mão detectada, são extraídos o número de dedos que se encontrem esticados e passada essa informação a um classificador com um algoritmo com três níveis. No primeiro são obtidos os gestos de treino que apresentam o mesmo número de dedos esticados. Caso o número de dedos esticados coincida com mais que um dos gestos de treino, são calculadas as direcções dos dedos e os ânglos formados entre eles que servirão de elemento de distinção. O sistema foi treinado para reconhecer nove gestos com a palma da mão virada para a câmara e apresentou uma precisão no reconhecimento de 88,72%.

3.4. Reconhecimento de gestos dinâmicos

O reconhecimento de gestos dinâmicos tem por objectivo extrair características de sequências de observações de forma a poder classificá-las inequivocamente. Como características extraídas destacam-se a forma da mão, a trajectória e informação

espacial. De seguida, são apresentadas algumas propostas retiradas do estado da arte para o reconhecimento de gestos dinâmicos isolados e contínuos de uma língua gestual.

Hernandez-Rebollar em [3] apresenta uma solução para captura e tradução de gestos dinâmicos isolados da ASL em voz e texto. Para a aquisição da informação das mãos e das articulações do corpo foram combinadas as luvas *AcceleGlove* com dois braços esqueletos. Os gestos adquiridos são segmentados em fonemas segundo a definição de Costello [45] pela decomposição das observações em movimentos e pausas. Para a classificação de gestos o sistema foi baseado em modelos de Markov-não observáveis e redes neuronais. O reconhecimento de gestos foi testado com 18 gestos executados com uma das mãos e atingiu 98% de reconhecimentos correctos. O sistema provou ser escalável sendo o léxico estendido para 176 gestos, executados com uma das mão, atingindo 94% de taxa de reconhecimento.

No trabalho de Thad Starner e Alex Pentland [40] é proposta uma solução onde o utilizador necessita de utilizar duas luvas coloridas para que seja possível seguir as mãos em tempo real através do processamento das imagens disponibilizadas por uma câmara RGB. O sistema não tenta produzir uma descrição muito detalhada da forma da mão mas sim produzir uma descrição "grosseira" da sua forma, orientação e trajectória. Esta informação servirá como entrada de um HMM utilizado para a classificação de frases (gestos dinâmicos) da ASL contendo 40 palavras. Para o conjunto de treino foram utilizadas 395 sequências e para o conjunto de teste foram utilizadas outras 99 sequências. Este classificador obteve uma percentagem de reconhecimentos correctos de 99,2% utilizando regras gramaticais para desambiguar classificações.

Christian Vogler e Dimitris Metaxas em [46] apresentaram uma proposta para realizar reconhecimento contínuo de gestos dinâmicos da ASL utilizando fonemas em vez de representar cada gesto como a unidade básica. A segmentação de fonemas é realizada através da decomposição das sequências em momentos de pausas. Como classificador apresentam uma solução baseada em HMMs. Os testes a esta solução, para 22 fonemas da ASL, apresentam uma percentagem de reconhecimentos correctos de 91,19%.

Zahoor Zafrulla et al. em [6] apresentam uma comparação de um sistema de reconhecimento de gestos dinâmicos da ASL, baseado no dispositivo Microsoft Kinect,

com o sistema CopyCat que recorre a luvas coloridas e acelerómetros para obter os movimentos da mão. Os dois sistemas tiram partido de modelos de Markov-não observáveis para classificar os gestos. O melhor resultado, 74,6% de reconhecimentos correctos foi obtido para palavras como unidade básica de gesto e com o executor permanecendo sentado. Os resultados obtidos revelaram aproximar-se dos 85,42% de precisão obtida pelo sistema CopyCat para gestos com o executor sentado.

No trabalho de Chao Sun et al. [27] é apresentada uma proposta de reconhecimento de fonemas da ASL com um algoritmo baseado em codificação de exemplos discriminativos. Esta proposta também recorre ao dispositivo sensorial de profundidade Microsof Kinect para a aquisição de informação. Os gestos de aprendizagem foram agrupados em contentores de classe onde cada contentor é descrito como um vector n-dimensional de semelhanças entre os n gestos de exemplo da classe. Para classificar uma classe são seleccionados os exemplos mais discriminativos do vector anteriormente calculado. Para a validação da proposta o sistema foi testado com uma base de dados criada pelos autores que contém 73 gestos da ASL. O sistema apresentou uma percentagem de reconhecimentos correctos de 85,5%.

No trabalho de Mahbub et al., "*A template matching approach of one-shot-learning gesture recognition*" [9], é proposta uma solução baseada na combinação de medidas estatísticas, transformações no domínio da frequência e representação do movimento obtidos através das imagens de profundidade disponibilizadas pelo dispositivo sensorial Microsoft Kinect. O método proposto inicialmente utiliza parâmetros extraídos das imagens de profundidade para separar gestos diferentes e removido o fundo através de um limiar de *threshold*. De seguida são extraídas características de gesto através de três operações básicas – calcular o desvio padrão entre imagens, obter transformadas bidimensionais de Fourrier e construir as imagens de histórico de movimento. Estes métodos são combinados para encontrar o método mais adequado para reconhecimento de gestos. Por último, concluíram que ao aplicar classificadores baseados no coeficiente de correlação obtido através do desvio padrão das transformadas bidimensionais de Fourier e utilizar as imagens de histórico de movimento em nada favoreceu o reconhecimento.

3.5. Reconhecimentos de gestos estáticos

O reconhecimento de gestos estáticos focam-se, de uma forma geral, em extrair características da mão predominante de forma a criar uma representação o menos ambígua possível. Como características extraídas destacam-se a forma da mão e outras medidas estatísticas. De seguida são apresentadas algumas propostas para o reconhecimento de gestos estáticos de uma língua gestual retiradas do estado da arte.

José M. Dias et al. no trabalho intitulado "*O.G.R.E. – Open Gestures Recognition Engine*" [10] apresentam uma proposta de solução para o reconhecimento de gestos estáticos e dinâmicos utilizando uma câmara a cores para a aquisição da informação de uma das mãos. Foi ainda criada uma forma de combinar gestos estáticos e dinâmicos denominada pelos autores de *Staged Paths*. De cada imagem é removido o fundo para que seja possível obter uma imagem da mão, calcular o seu contorno e derivar métricas que o caracterizam. O reconhecimento de gestos estáticos é realizado através de *template matching* das imagens da mão ou através da análise dos vectores de características. Para o reconhecimento de gestos dinâmicos analisam formas geométricas básicas produzidas pela trajectória da mão, por exemplo um triângulo, um círculo ou um quadrado, com recurso a lógica difusa, que é normalmente utilizada para reconhecer formas de caligrafia. Os testes realizados ao sistema para reconhecer gestos do alfabeto da LGP, mostram que os melhores resultados foram obtidos pelos métodos de *template matching* e o de análise de geometria da mão, onde foram obtidas percentagens de reconhecimentos correctos de 53,6% e 58,1% respectivamente.

A dissertação de Mestrado de Rui Almeida [11], focou-se no reconhecimento de gestos isolados estáticos e dinâmicos da LGP. Para a aquisição da informação foi adoptado o sensor de profundidade Microsoft Kinect. No que diz respeito à classificação de gestos estáticos foram implementados dois algoritmos que utilizam as imagens de profundidade da mão após normalização para classificar 26 letras do alfabeto da LGP. O primeiro algoritmo baseia-se na análise das coordenadas máximas e mínimas a duas dimensões do conjunto de gestos e obteve uma precisão global de 89%. O segundo algoritmo é baseado no método de *template matching* que classifica com base em interpolações lineares permitindo calcular a forma média da mão de uma classe sendo

obtida uma precisão global de reconhecimentos correctos de 100%. Para classificar gestos dinâmicos foram implementados dois algoritmos que se baseiam no caminho percorrido pela mão durante a execução de um gesto para a classificar 14 gestos. No primeiro algoritmo são analisados os caminhos percorridos pelas mãos para obter a menor distância entre eles. Este algoritmo obteve uma precisão de 100%. Como segundo algoritmo para classificar gestos dinâmicos, apresentam um HMM para estimar o caminho percorrido pela mão durante a execução de um gesto, tendo sido obtida uma percentagem global de reconhecimentos correctos de 58%.

No trabalho "*Robust Hand Gesture Recognition with Kinect Sensor*" [8], Zhou Ren et al. apresentam uma proposta de reconhecimentos de gestos com base no algoritmo de *Finger-Earth Movement Distance* por eles apresentado anteriormente. Para a detecção e extracção da forma da mão são utilizadas a imagens de profundidade e de cor que o dispositivo Microsoft Kinect disponibiliza que por sua vez são processadas para extrair o contorno dos dedos. O contorno é "recortado" de forma a obter a zona que contém os dedos que estão esticados. De seguida o contorno obtido através do passo anterior é representado através de uma série temporal onde para o reconhecimento é calculado o menor deslocamento necessário para alinhar duas destas séries. Para o conjunto de dez gestos referentes aos números de zero a nove, com a palma da mão virada para a frente, este sistema apresentou uma precisão média de 90,6%.

No trabalho de Trong-Nguyen e Huu-Hung Huynh em [36], é proposta uma solução para reconhecimento de gestos estáticos da ASL baseada na análise de componentes principais e numa rede neuronal artificial. O processo inicia pela detecção da mão pré-processando as imagens obtidas através de uma câmara a cores. De seguida são extraídas características das imagens através do cálculo dos valores e vectores próprios que servirão como entradas da rede neuronal. Os testes a esta solução apresentaram uma percentagem média de reconhecimentos correctos de 94,3%.

Na dissertação de Mestrado de Ana Paula Sousa [12], é apresentada uma proposta de reconhecimento de gestos estáticos isolados da LGP com recurso ao dispositivo Microsoft Kinect. Após a aquisição das imagens de profundidade da mão, é criado um vector de características discriminativas da sua posição e do seu contorno. Para a

classificação foi utilizada uma ferramenta que disponibiliza vários classificadores e no melhor dos casos foi obtida uma percentagem de classificações correctas de 96,69% com um algoritmo baseado no vizinho mais próximo. Como segundo melhor resultado foi obtida uma percentagem de classificações correctas de 94,29% com um algoritmo baseado em árvores de decisão.

4. Arquitectura do Sistema

Como referido anteriormente, pretende-se contribuir com uma solução que realize o reconhecimento e tradução contínua de gestos em tempo real aplicada à Língua Gestual Portuguesa. A solução não deve ter apenas em conta gestos estáticos (não requerem movimento) mas também gestos dinâmicos. Para dar suporte aos requisitos acima referidos é proposta a arquitectura que pode ser dividida em quatro módulos com funções distintas, apresentada na Figura 5.

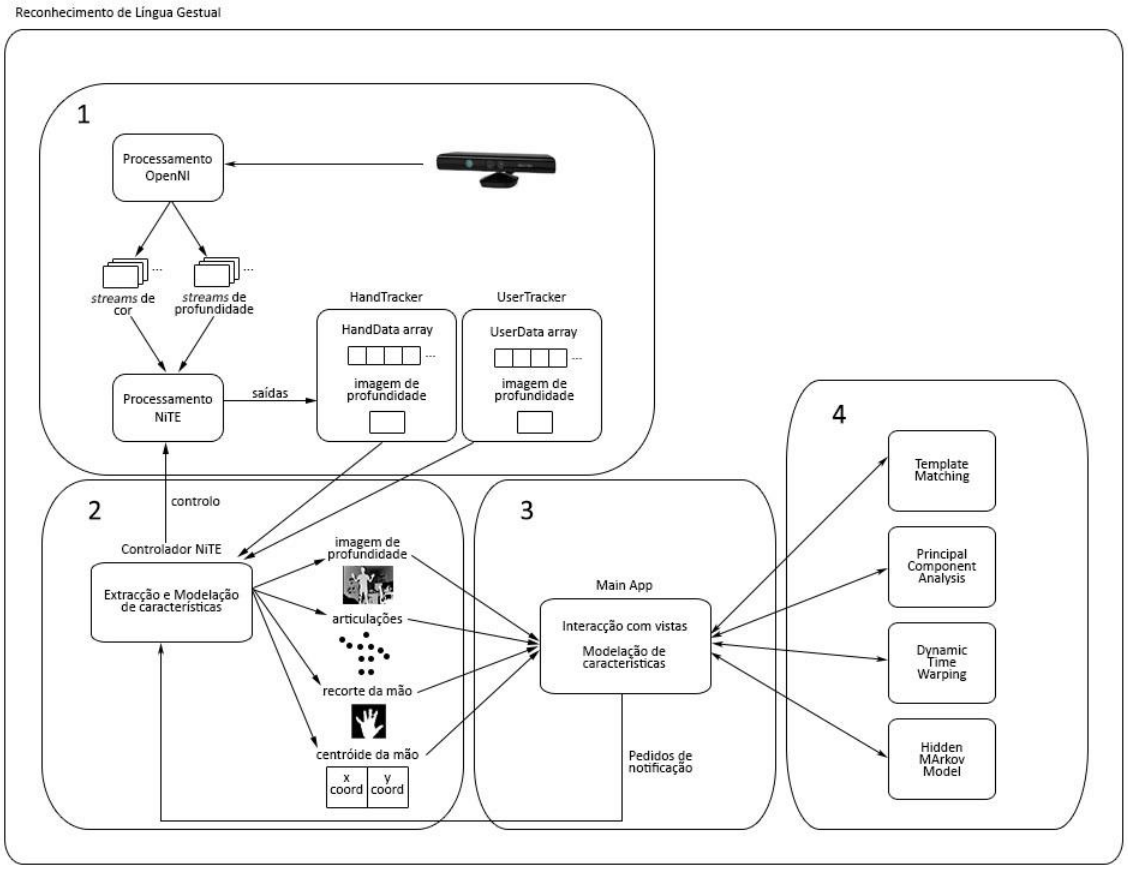


Figura 5 - Arquitectura do sistema proposto

Assumindo cada módulo como um bloco, o primeiro destina-se ao acesso ao dispositivo sensorial, ao pré-processamento e modelação da informação obtida para que seja representada perante o sistema. O segundo módulo é responsável pelo acesso e controlo das operações realizadas pelo primeiro módulo. É também neste módulo que é realizado o processamento e modulação da informação para extrair as características

necessárias a partir da informação disponibilizada pelo primeiro módulo. Após a extracção das características, este módulo tem a responsabilidade de notificar os seus subscritores que determinas características foram extraídas e estão disponíveis para serem utilizadas. O terceiro módulo é responsável pela criação da interface de utilização e toda a interacção com ela. Este módulo tem ainda a responsabilidade de requisitar a informação necessária ao segundo módulo e por sua vez utilizá-la consoante as necessidades, ou seja, para apresentação ou para a fornecê-la aos algoritmos de reconhecimento definidos no quarto módulo. Por último, o quarto módulo é responsável por todo o comportamento relacionado com os algoritmos de reconhecimento implementados.

4.1. Sensor e acesso à informação

O primeiro passo de um sistema de reconhecimento é a aquisição da informação. Uma vez que os componentes mínimos da LGP são manuais, como descrito na secção 2.2.1, revela-se imperativo a aquisição do detalhe da mão e a sua relação com o corpo.

Numa primeira abordagem foi testado o dispositivo Leap Motion, apresentado na secção 2.3.2. Este dispositivo foi desenvolvido para fornecer exclusivamente informação a três dimensões das mãos e dos dedos. É portanto capaz de fornecer as coordenadas x, y, z da mão, a rotação em torno dos três eixos, *pitch*, *yaw* e *roll*, respectivamente, a magnitude da mão que representa a distância Euclidiana entre a origem e o ponto x, y, z, a direcção da palma da mão, o raio da esfera da mão e a colecção de dedos associados a uma mão detectada. Relativamente aos dedos é também possível saber as coordenadas x, y, z das pontas dos dedos detectados, a rotação em torno dos três eixos, a magnitude, o comprimento e a largura estimados em milímetros [32]. Após vários testes chegou-se à conclusão que mesmo com um elevado *frame rate* deste sensor (300 fps), o factor que impossibilitou a utilização deste sensor, tal como descrito por Terdiman [47], foi o curto campo de visão não ser satisfatório para soluções em que as mãos navegam para além de 600mm de distância do sensor. Para além disso, a precisão apresentada por este sensor no que toca ao detalhe dos dedos fica muito aquém do

esperado, pois a configuração da mão e a perspectiva que o sensor tem dela, faz com que falhe na detecção dos dedos em diversas situações.

Numa segunda abordagem foi testado o dispositivo Microsoft Kinect, apresentado na secção 2.3.1, para a aquisição da informação a partir das imagens de profundidade disponibilizadas por este sensor. Para o acesso ao sensor e às imagens de profundidade foram testadas duas *frameworks*, Microsoft SDK (MS SDK) [1] e OpenNI [31]. Estas duas *frameworks* podem coexistir a partir da versão 2.x do OpenNI pois já permite utilizar os drivers da Microsoft para acesso ao sensor. As duas revelaram desempenhar o pretendido embora o MS SDK seja de código fechado e desenvolvido para suportar apenas o dispositivo sensorial Kinect, o que não acontece com o OpenNI. Esta segunda *framework* é de código aberto e foi desenvolvida para ser independente do sensor utilizado, comunicando não só com o Kinect mas também com outros sensores, tais como PrimeSense ou ASUS. Por outro lado, o MS SDK fornece nativamente a informação de até vinte articulações de cada utilizador detectado enquanto que o OpenNI não fornece esta informação por si só. Contudo, complementado com o *middleware* NiTE, é possível obter a mesma informação. Este *middleware* acede às imagens fornecidas pelo OpenNI e processa-as internamente para conseguir capturar o movimento e o posicionamento de até vinte articulações do corpo.

Na tentativa de obter maior detalhe da mão a partir do Kinect, foram realizados testes à *framework* Kinect 3D Hand Tracking [48], que segundo a equipa que a criou, é capaz de estimar 26 graus de liberdade (*degrees of freedom*, DoF). Esta *framework* apresentou o problema de ter dependências de uma versão antiga do OpenNI 1.x e do NiTE 1.x onde é necessário trocar os drivers do Kinect consoante a solução que se tem (SensorKinect para as aplicações do OpenNI e os drivers da Microsoft para as aplicações que utilizam o MS SDK). Após alguns testes, verificou-se que a inicialização da detecção da mão não é fácil para o utilizador final e mesmo que se consiga iniciar a detecção, a mão e os dedos são perdidos com grande facilidade, não se revelando uma solução viável para o problema.

A escolha do sensor a utilizar para a aquisição da informação incidiu sobre o dispositivo sensorial Kinect uma vez que revelou maior probabilidade de sucesso para o

desenvolvimento do sistema pretendido. Para o acesso ao sensor optou-se por utilizar a combinação da *framework* OpenNI juntamente com o *middleware* NiTE por ser de código aberto e por suportar o acesso a vários sensores. Esta última razão permite que, com o surgir de novos dispositivos seja possível adoptá-los para a solução sem grande esforço de implementação, desde que sejam compatíveis com o OpenNI.

4.2. Classificadores e reconhecimento

Conforme descrito anteriormente, o sistema deverá ser capaz de identificar gestos estáticos e gestos dinâmicos da Língua Gestual Portuguesa. Face aos requisitos foi decido implementar dois algoritmos de reconhecimento, presentes na literatura, para gestos estáticos e dois para gestos dinâmicos. Para os gestos estáticos foram escolhidos dois algoritmos clássicos de reconhecimentos com base em observações da mão, *template matching* e análise de componentes principais. O primeiro algoritmo foi escolhido por ser simples de implementar, uma vez que utiliza directamente os padrões de exemplo. Com o aumento do número de padrões e consequente aumento do tempo necessário para classificação, o reconhecimento em tempo real pode tornar-se impraticável. Por esta razão foi decido adoptar um segundo algoritmo, análise de componentes principais, que permite a redução do número de características necessárias para o reconhecimento. Para o reconhecimento de gestos dinâmicos foram adoptados dois algoritmos que permitem o reconhecimento com base em sequências de observações temporais. O primeiro algoritmo adoptado foi o alinhamento temporal dinâmico por ser extremamente eficiente em medir semelhanças em séries temporais, minimizando os efeitos de deslocamento e distorção temporal. No entanto, tal como no algoritmo de *template matching*, com o aumento do número de gestos de exemplo pode levar a que este algoritmo seja ineficiente para reconhecimento em tempo real. O segundo algoritmo adoptado foi o modelo de Markov não-observável. Este algoritmo tem tido bastante aplicabilidade em sistemas de reconhecimento de gestos e permite reduzir o número de dimensões eliminando parcialmente o problema do aumento linear do tempo de processamento durante a classificação.

4.3. Ferramentas de desenvolvimento

Como ambiente de desenvolvimento foi adoptado o Microsoft Visual Studio 2013 e utilizou-se a linguagem de programação C# na versão .Net Framework 4.5. Para o acesso ao sensor Kinect, como referido no secção 4.1, optou-se por utilizar a combinação da *framework* OpenNI v2.2.0.33 [31] juntamente com o *middleware* NiTE v2.2.0.11. Uma vez que estas duas *frameworks* não têm suporte para C# foram utilizados dois *wrappers* desenvolvidos por Soroush Falahati [49], NiWrapper.Net e NiWrapper.NiTE.Net, para que fosse possível o acesso às referidas *frameworks*. Para processamento de imagem foi utilizada a *framework* AForge.Net v2.2.5.0 [50]. No que diz respeito a processamento estatístico de informação, algoritmos de aprendizagem e reconhecimentos de padrões foi utilizada a *framework* Accord.Net v2.12.0 [51] desenvolvida sobre a *framework* AForge.Net.

4.4. Interface de utilização

O protótipo cliente está organizado numa única janela tal como apresentado na Figura 6. Do lado esquerdo estão presentes os controlos de visualização da informação obtida do NiTE. Ao centro é apresentada a informação de captura consoante a selecção do lado esquerdo. Esta secção é também utilizada para visualizar os gestos presentes em memória dos vários algoritmos de reconhecimento. Do lado direito foram definidos quatro separadores contendo as opções de controlo e de visualização de cada algoritmo de reconhecimento implementado.

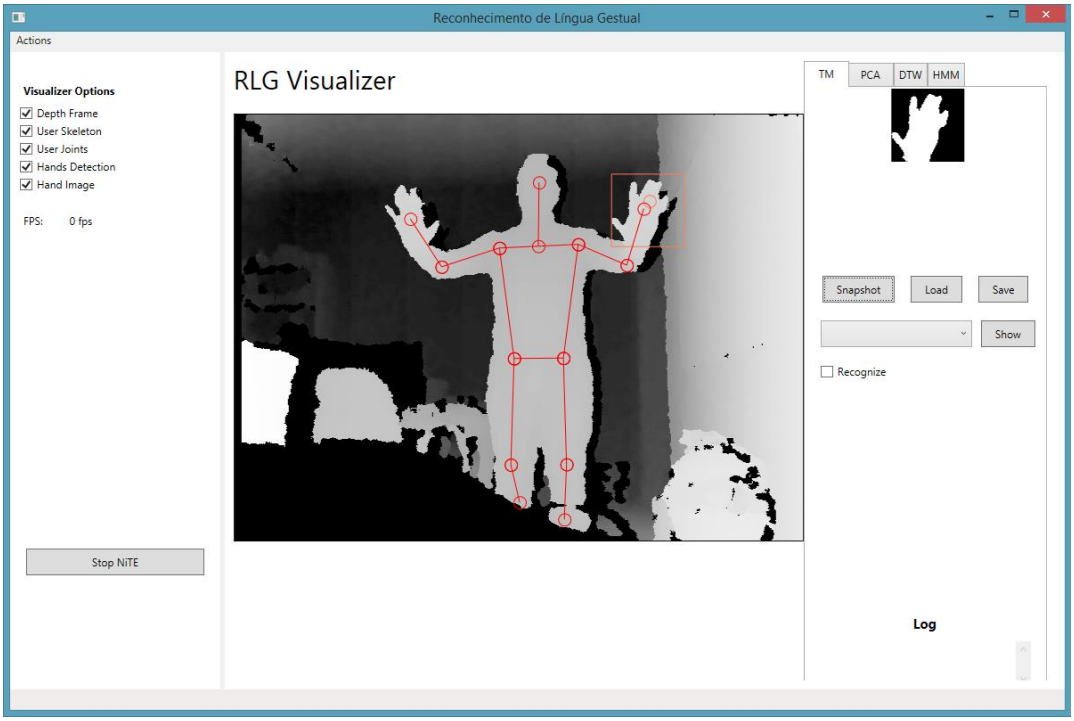


Figura 6- Interface de utilização

A interface de utilização foi desenvolvida com recurso ao subsistema gráfico Windows Presentation Foundation (WPF) [52]. Foi adoptado o padrão Model-View-ViewModel (MVVM) cujo modelo de comunicação é apresentado na Figura 7. Este padrão promove a separação do desenvolvimento da interface gráfica da lógica de negócio. Uma grande vantagem deste padrão é utilizar *data bindings*, ou seja, nos componentes da View são declarados *bindings* de acesso às propriedades do View Model. Por sua vez, o View Model é responsável por expor uma representação dos objectos do Model para que sejam facilmente geridos e consumidos.

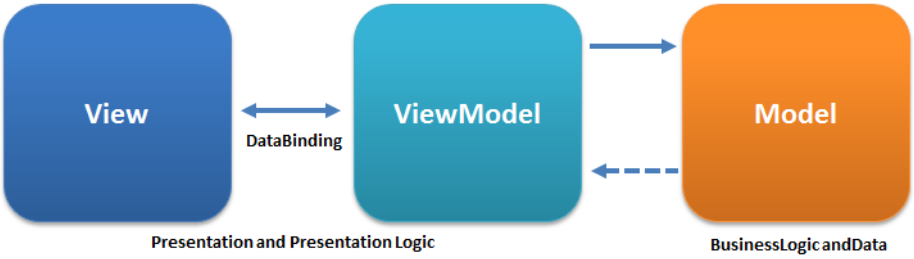


Figura 7 - Padrão MVVM (retirado: Wikipédia)

O WPF contém também um mecanismo de comandos que tem como principal objectivo separar a semântica e o objecto que o invoca, da lógica que o executa. Desta forma é possível definir comandos no View Model onde um ou mais componentes da View estão *bounded*, ou seja, é possível definir no Model uma implementação concreta para o comando sem existir ligação ao chamador. Outra grande vantagem da utilização de comandos é permitirem habilitar ou desabilitar componentes da View consoante o seu estado, se podem ou não ser executados.

5. Desenvolvimento

Neste capítulo é introduzida e apresentada em detalhe a abordagem adoptada para o desenvolvimento do sistema proposto. Inicialmente é apresentada a forma de acesso ao dispositivo sensorial Kinect e de seguida é apresentada a forma de extrair as características necessárias para o reconhecimento. São ainda apresentadas as implementações dos algoritmos de classificação de gestos estáticos e dinâmicos e a forma de os utilizar e comunicar com cada um deles. Por último é apresentada a base de dados criada para este trabalho que contém gestos estáticos e dinâmicos da LGP utilizados posteriormente para testar a solução proposta.

5.1. Acesso ao sensor Kinect e pré-processamento NiTE

Como apresentado na secção 4.1, foi adoptado o dispositivo sensorial Kinect para captura de informação de profundidade e utilizada a *framework* OpenNI para o acesso ao sensor. Após a aquisição das imagens de profundidade, apresentada na Figura 8, existe a necessidade de processar cada uma para extrair a informação que caracterize o corpo e a mão do utilizador. Como ferramenta auxiliar de pré-processamento e extracção de características a partir da imagem de profundidade, o sistema recorre ao *middleware* NiTE. Uma vez que não existe necessidade de aceder directamente ao OpenNI, mas sim à informação já processada pelo NiTE, apenas é necessário inicializar explicitamente o NiTE, ficando este encarregue de inicializar o OpenNI. No espaço de nomes do NiTE estão também definidas duas classes HandTracker e UserTracker, responsáveis por obter as imagens de profundidade e processá-las para extrair a informação das mãos e do corpo, respectivamente. Uma vez que o processamento executado por cada uma destas classes é uma operação computacionalmente dispendiosa, cada instância destas classes deve ser gerida de forma independente consoante as necessidades. Por último, cada uma destes pré-processadores tem definido um *delegate* no qual outros componentes se devem registar para receber de forma assíncrona a informação após o processamento.

Figura 8 - Imagem de profundidade

5.2. Extracção e processamento de características

Como referido na secção 4.1, no segundo módulo foi implementado um controlador e processador de informação proveniente do NiTE onde foram definidas quatro funções. A primeira é referente à inicialização e à comunicação com o objecto NiTE. A segunda é referente à decisão quanto à necessidade dos dois processadores de características, HandTracker e UserTracker, estarem a processar as imagens provenientes do NiTE. A terceira função é referente à modelação e extracção de características e por último, este módulo é responsável pela notificação dos seus subscritores que determinadas características foram processadas e estão disponíveis. Relativamente à extracção de características este controlador retorna informação de uma das mãos do utilizador e das suas articulações, isto porque, o sistema apenas permite a interacção com um utilizador de cada vez. Da mão são extraídas, para além do posicionamento, uma imagem binária rectangular e normalizada relativamente à distância contendo apenas a mão do utilizador. Relativamente às articulações do utilizador, este controlador devolve as coordenadas a 3 dimensões de cada articulação. Permite ainda obter a imagem total de profundidade que deu origem a cada informação extraída. O UML da definição desta

classe é apresentado na Figura 9. Neste UML foram removidos alguns atributos privados de classe menos relevantes por motivos de optimização de espaço.

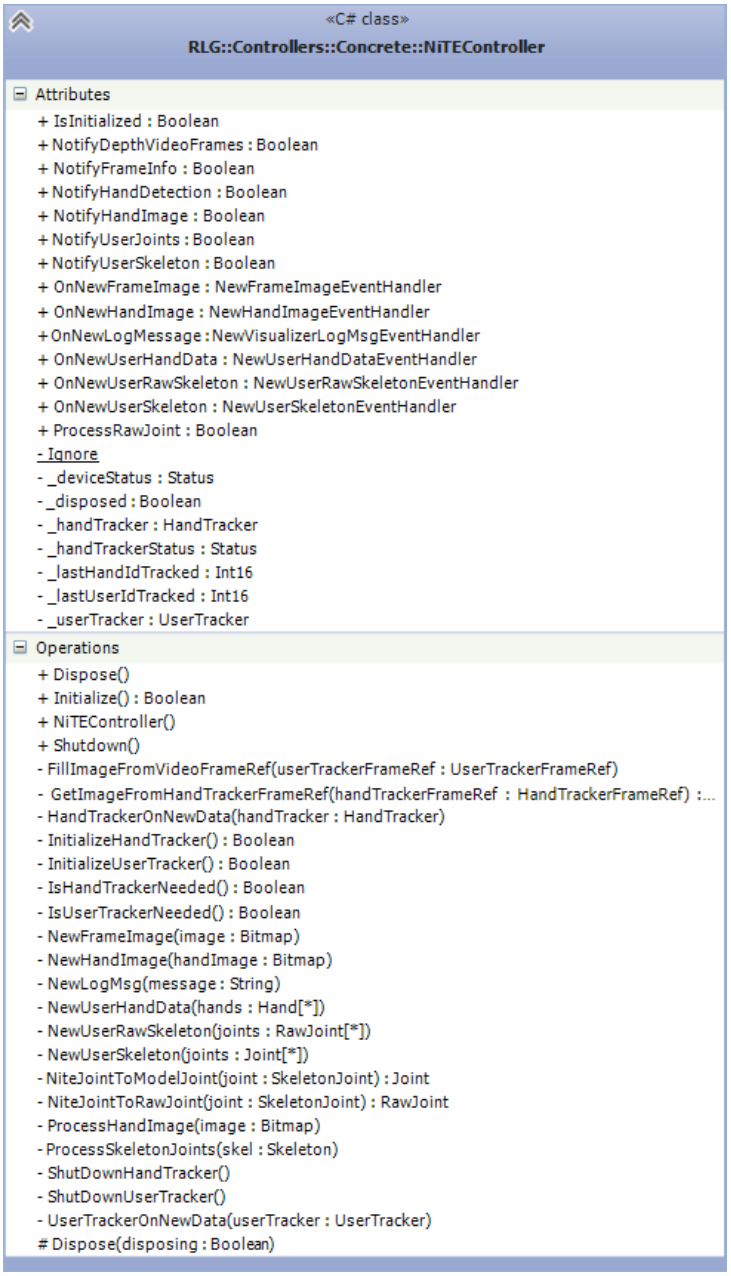


Figura 9 - UML da classe NiTEController

5.2.1. Extracção de características da mão

Para pré-processamentos das características da mão o sistema recorre ao *middleware* NiTE que disponibiliza o objecto HandTracker para o efeito. Este objecto é instanciado e inicializado apenas quando a informação da mão é necessária para o sistema, sendo

também subscrito o evento OnNewData para recepção de forma assíncrona da informação. O processo de inicialização é apresentado na Listagem 1. A partir deste momento, cada vez que o NiTE processar uma nova imagem irá notificar este controlador com a chamada ao método HandTrackerOnNewData.

```
/// <summary>
/// Initialize HandTracker object and attatch new data event
/// </summary>
/// <returns>If HandTracker is initialized</returns>
private bool InitializeHandTracker()
{
    if (this._deviceStatus != NiTE.Status.Ok) return false;
    if (this._handTracker == null)
        this._handTracker = HandTracker.Create();
    if (this._handTracker != null && this._handTracker.IsValid)
    {
        _handTrackerStatus =
this._handTracker.StartGestureDetection(GestureData.GestureType.Hand
Raise);
        if (_handTrackerStatus != NiTE.Status.Ok)
        {
            this._isHandTrackerInitialized = false;
            NewLogMsg(String.Format("{0} HandTracker initialyze
Error: {0}", _tag, _handTrackerStatus.ToString()));
            return false;
        }
        _handTracker.OnNewData += this.HandTrackerOnNewData;

        return this._isHandTrackerInitialized = true;
    }
    return this._isHandTrackerInitialized = false;
}
```

Listagem 1 - Inicialização do HandTracker

A informação proveniente do HandTracker é utilizada por este controlador para modelar as coordenadas do centro de massa da mão num objecto mais simples contendo as coordenadas a duas dimensões. Uma vez que o HandTracker retorna as coordenas x, y, z no sistema de coordenadas do dispositivo Kinect, apresentado na secção 2.3.1, as coordenadas do centro de massa da mão são convertidas para o sistema de coordenadas 2D do monitor onde são representadas pela distância ao ponto superior esquerdo da imagem total. A conversão destas coordenadas é realizada à custa das equações (15) e (16).

$$sceen.x = \frac{world.x}{world.z} * zoom + center.x \quad (15)$$

$$sceen.y = \frac{world.y}{world.z} * zoom + center.y \quad (16)$$

Caso este posicionamento esteja a ser requisitado, são notificados os subscritores do evento OnNewUserHandData que uma nova imagem foi processada e que as coordenadas da mão estão disponíveis.

Para além de serem obtidas as coordenadas do centro de massa da mão foi implementado um algoritmo para extrair, da imagem total de profundidade, uma nova imagem que incluirá a mão centrada nela, normalizada em relação à distância em z do Kinect. A sequência de operações aplicadas a cada imagem é apresentada na Figura 10.

Figura 10 - Processamento de imagens da mão

Em primeiro lugar é necessário calcular a dimensão e posicionamento da janela delimitadora na imagem total. Para tal, como ponto de referência são consideradas as coordenadas do centro de massa obtidas anteriormente e de seguida aplicada uma margem nas duas direcções. Inicialmente são calculadas as coordenadas do topo superior esquerdo do recorte e de seguida calculados os deslocamentos a aplicar nas duas direcções. É ainda importante referir que o cálculo desta margem tem em conta a distância em z para que seja possível criar um factor de escala da janela e obter uma imagem com a mesma proporção da mão independentemente da distância. Após aplicado o recorte é então obtida uma imagem de profundidade rectangular com a mão centrada nela. Alguns exemplos são apresentados na Figura 11.

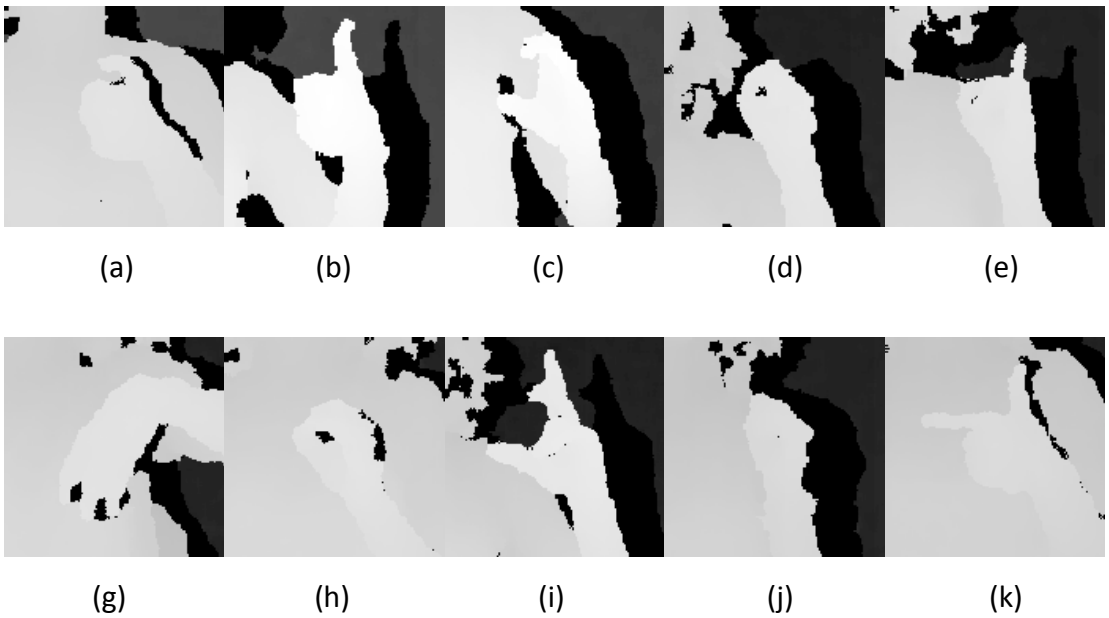


Figura 11 - Imagens de recorte da mão;

(a) – Número 5; (b) – Letra B; (c) – Letra C; (d) – Letra G; (e) – Letra I; (g) – Letra M; (h) – Letra O ou número 0; (i) – Letra R; (j) – Letra S; (k) – Letra T;

Uma vez que estas imagens vêm representadas em escalas de cinzentos, onde os tons mais claros representam o que está mais perto e os tons mais escuros o que se encontra mais longe, é possível remover o fundo da imagem de forma a que apenas a mão permaneça na imagem. Para este processo o controlador recorre à biblioteca

AForge.Imaging.Filters que contém vários filtros para processamento de imagem. Inicialmente é aplicado um filtro de intensidade que recebe como parâmetros os valores máximos e mínimos a manter. Desta forma, todas as intensidades que estejam fora deste intervalo são passadas ao valor zero. Para calcular os limites a aplicar no filtro, uma vez que a imagem contém a mão centrada nela, é obtido o valor do *pixel* do ponto de central da imagem que se espera ser da mão e aplicada uma margem de segurança de intensidade de -5 para o limite inferior e +5 para o superior. O resultado da aplicação do filtro é apresentado na Figura 12.

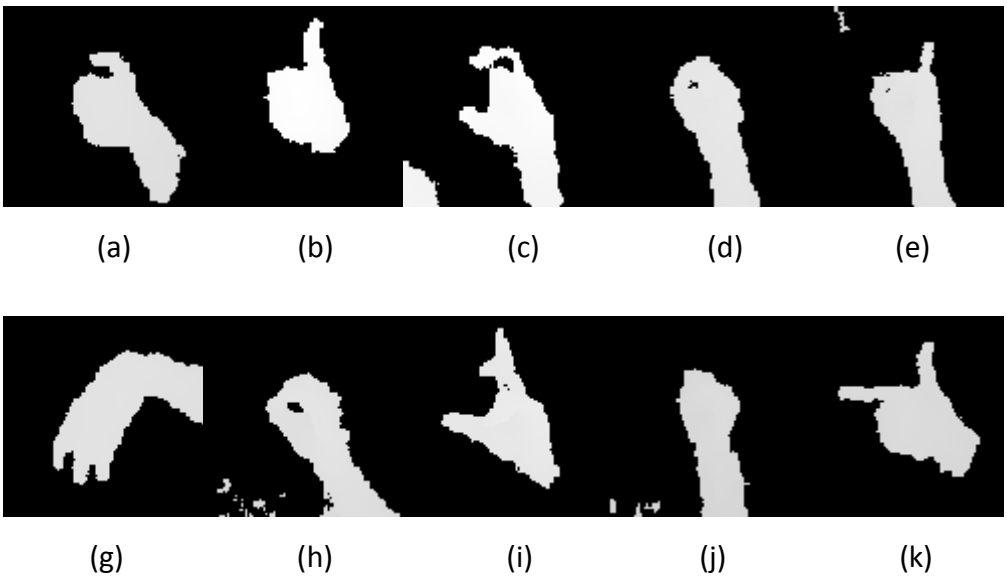


Figura 12 - Imagem da mão após ser aplicado o filtro de intensidade;

(a) – Número 5; (b) – Letra B; (c) – Letra C; (d) – Letra G; (e) – Letra I; (g) – Letra M; (h) – Letra O ou número 0; (i) – Letra R; (j) – Letra S; (k) – Letra T;

Para normalizar as imagens em relação à distância da mão ao sensor, cada *pixel* da imagem é comparado com um limiar (*threshold*), sendo colocado a preto (0) ou branco (1 ou 255), conforme o seu valor seja menor ou maior que o limiar, respectivamente. O valor definido para este limiar foi de 127 que representa o valor intermédio da escala de cinzentos, representada para 8 bits. Para eliminar possíveis falhas de preenchimento no interior da região da mão é aplicado um filtro denominado de *FillHoles*, presente na biblioteca AForge.Imaging.Filters, de modo a preencher possíveis falhas com nível

branco e obter a região totalmente preenchida. Como o filtro de intensidade, inicialmente aplicado, tem por base um limite inferior e superior de intensidade, foi detectado que por vezes permanecem na imagem outras regiões do corpo que se encontram à mesma distância que a mão e consequentemente a sua distância encontra-se dentro do intervalo definido para o filtro. Para conseguir minimizar este problema é ainda obtida a maior região conexa presente na imagem, que representará a região da mão, com recurso ao filtro ExtractBiggestBlob também presente na biblioteca Aforge.Imaging.Filters. Após obtida a maior região conexa é criada uma nova imagem, do tamanho da imagem inicial, totalmente preenchida com nível preto e sobreposta a maior mancha obtida na posição inicial. O resultado final destas operações é apresentado na Figura 13.

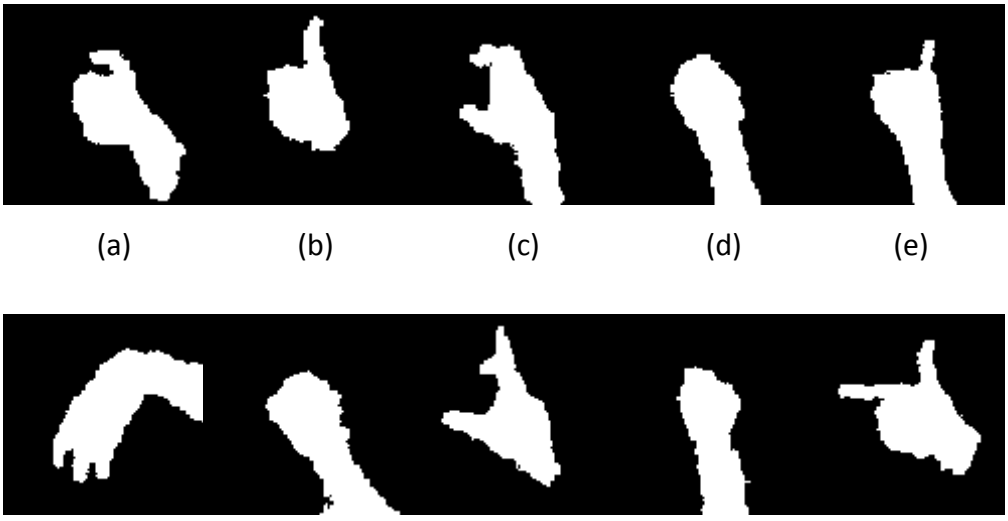


Figura 13 - Imagem da mão após processamento;

(a) – Número 5; (b) – Letra B; (c) – Letra C; (d) – Letra G; (e) – Letra I; (g) – Letra M; (h) – Letra O ou número 0; (i) – Letra R; (j) – Letra S; (k) – Letra T;

Outros módulos do sistema que necessitem de obter estas imagens terão que subscrever o evento OnNewHandImage para serem notificados assincronamente que a extracção de uma imagem da mão foi concluída e está pronta para ser utilizada.

(A) Tentativa de criar um esqueleto da mão

Para obter maior detalhe na representação da mão, os algoritmos que tentam calcular as coordenadas das pontas dos dedos a partir da imagem, sofrem do mesmo problema apresentado para o dispositivo Leap Motion. Isto é, configuração da mão e a perspectiva que o sensor tem dela, faz com que a detecção dos dedos falhe em inúmeras situações. Por esta razão, estas características não são consideradas fiáveis para o reconhecimento de uma língua gestual. Como alternativa foi testada a hipótese de gerar um esqueleto da mão onde, independentemente da configuração da mão, fosse obtida uma representação. O reconhecimento de gestos estáticos e dinâmicos baseados em esqueletos foi alvo de estudo sendo apresentada uma proposta com base no histograma de orientação por Bogdan Ionescu et al [53]. Posto isto, foi implementado um algoritmo para criar um esqueleto representativo da configuração da mão, apresentado na Figura 14, com base nas imagens de recorte da mão que o controlador disponibiliza.

Figura 14 – Imagem inicial mão e esqueleto gerado

Para criar este esqueleto foi aplicado o operador morfológico *thinning* [54] de forma sequencial, usualmente aplicado a imagens binárias. Dada uma imagem e um elemento estruturante permite remover *pixeis* em primeiro plano até que todas as linhas da imagem tenham somente um *pixel* de espessura de forma a criar um esqueleto.

Após alguns testes a este algoritmo chegou-se à conclusão que esta operação não é computacionalmente admissível para reconhecimento em tempo real porque o tempo necessário para aplicar o operador *thinning* a cada imagem é demasiado excessivo.

5.2.2. Extracção de características do corpo

Quando existe a necessidade de obter a informação das articulações do corpo, o controlador recorre ao objecto *UserTracker*. A inicialização e instanciação deste objecto segue o mesmo princípio do objecto *HandTracker*, ou seja, apenas é instanciado e inicializado quando a informação que fornece é necessária. A partir do momento em que evento *OnNewData* é subscrito no objecto *UserTracker*, cada vez que uma nova imagem for processada, o controlador irá ser notificado de forma assíncrona com a chamada ao método *UserTrackerOnNewData*. A cada notificação recebida, a informação é utilizada para extrair as articulações do primeiro utilizador detectado. Ainda a partir do objecto *UserTracker* é possível obter uma referência para o objecto *UserTrackerFrameRef* que contém a informação pré-processada da imagem de profundidade. A partir deste ponto, caso existam pedidos de notificação das articulações do utilizador, é criada uma lista de objectos "*Joint*" contendo apenas as coordenadas x, y, z representada no referencial do Kinect e o nome de cada articulação detectada. Uma representação das articulações processadas por este controlador é apresentada na Figura 15.

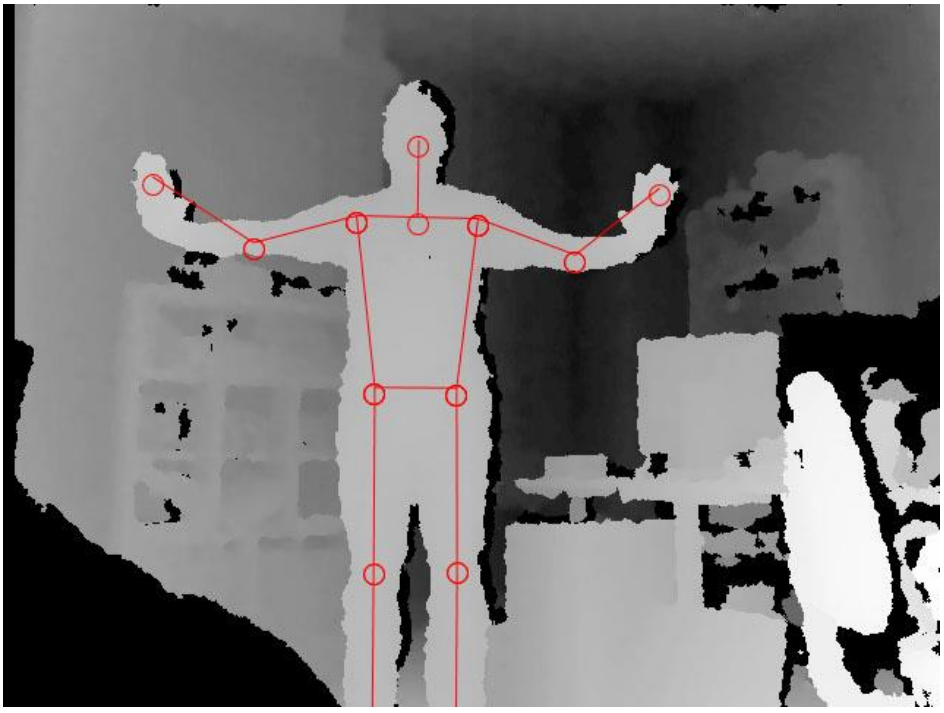


Figura 15 - Esqueleto e articulações

Por último, os subscritores do evento *OnNewUserSkeleton* que estejam à espera de receber a informação dos esqueletos extraídos por este controlador são notificados. Quando existe a necessidade de obter a imagem total de profundidade o controlador recorre também ao objecto *UserTracker* para obter essa imagem, convertendo-a para Bitmap e notificando os subscritores do evento *OnNewFrameImage*.

Cada uma das características processadas pelo controlador é obtida de forma independente, quer isto dizer, que apenas é extraída e modelada a informação que no momento esteja a ser requisitada evitando assim processamentos desnecessários. A partir do momento em que não existe a necessidade de obter informação proveniente do *UserTracker*, também para eliminar o número de processamentos desnecessários, é interrompida a detecção, removida a subscrição do evento *OnNewData* e destruído o objecto. A execução desta operação é apresentada na Listagem 2.

```
/// <summary>
/// Shutdown UserTracker object
/// </summary>
private void ShutDownUserTracker()
{
    this._isUserTrackerInitialized = false;
    if (this._userTracker == null) return;
    if (this._lastSeenUserId > -1)
    {

this._userTracker.StopSkeletonTracking(this._lastUserIdTracked);
        this._lastUserIdTracked = -1;
    }
    this._userTracker.OnNewData -= this.UserTrackerOnNewData;
    this._lastUserIdTracked = -1;
    this._userTracker.Destroy();
    this._userTracker = null;
}
```

Listagem 2 - Destrutor do objecto UserTracker

5.3. Classificação e reconhecimento de gestos estáticos

Para a classificação e reconhecimento de gestos estáticos foram implementados dois algoritmos, *template matching* e análise de componentes principais. Os dois algoritmos recebem como características as imagens binárias da mão já processada com o

algoritmo apresentado no capítulo anterior. Uma vez que o algoritmo de extracção das imagens da mão poderá devolver imagens com dimensões diferentes, para diminuir o número de imagens de exemplo necessárias para a classificação, os algoritmos desenvolvidos aplicam um escalamento para uma dimensão pré-definida a todas as imagens que processam. Nos dois algoritmos, para além de ser possível obter a classificação dada uma nova imagem, foi implementado um mecanismo de reconhecimento temporal para aumentar a probabilidade de acerto no reconhecimento em tempo real, ou seja, uma sequência de n classificações da mesma letra é um forte indício de que o gesto corresponde a essa letra. Por outro lado, as transições entre gestos diferentes tendem a gerar sequências de classificações diferentes, sendo possível desta forma minimizar o número de classificações erradas.

5.3.1. *Template Matching*

Como apresentado na secção 3.1.1, os algoritmos de *template matching* correlacionam os padrões de exemplo com um novo padrão de forma a determinar um grau de semelhança entre eles. Os padrões fornecidos a este algoritmo são as imagens da mão já processadas pelo algoritmo apresentado na secção 5.2. Para a concretização deste algoritmo foi implementada a classe *TemplateMatchingRecognizer* cuja representação UML é apresentada na Figura 16. Nesta classe foram definidas as operações elementares de adicionar e remover uma imagem de exemplo, de obter a classe mais semelhante dada uma nova imagem e de ler ou exportar as imagens de exemplo de/para ficheiro. As imagens de exemplo são guardadas num contentor de objectos *TmClassRecognizer* que agrupa as imagens de exemplo de uma determinada classe. A representação UML desta última classe é apresentada na Figura 17.

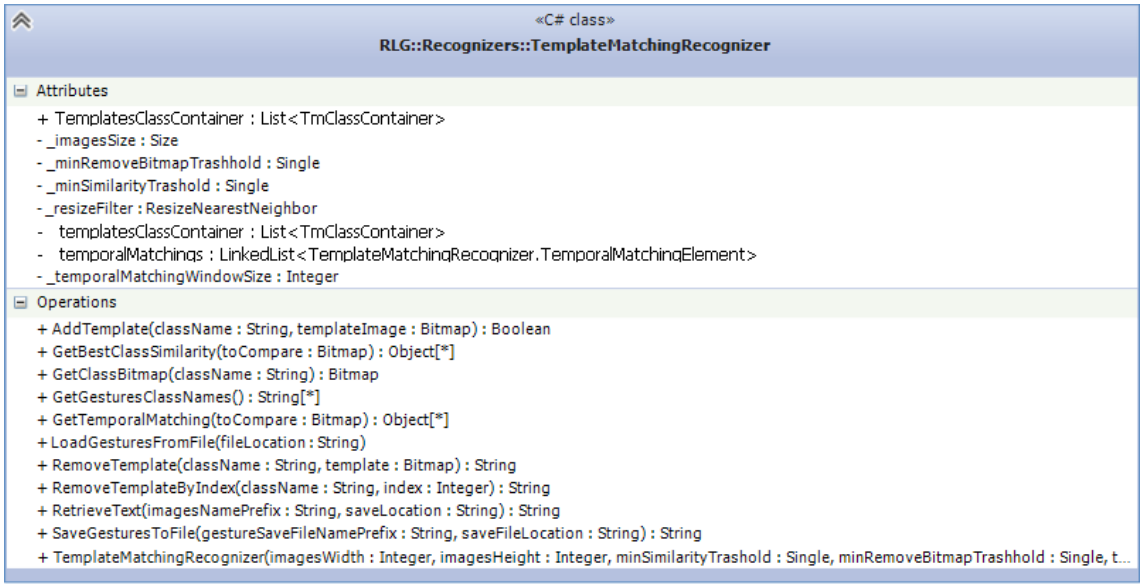


Figura 16- UML da classe TemplateMatchingRecognizer

Para adicionar uma nova imagem de exemplo a classe *TemplateMatchingRecognizer* disponibiliza o método *AddTemplate* que recebe como parâmetro o Bitmap da imagem a adicionar e a classe a que pertence. De seguida é pesquisada a lista de contentores para verificar se já se encontra presente um contentor para a classe a adicionar e caso não seja encontrado, é então criado um contentor para a nova classe sendo de seguida adicionado à lista. Por último é invocado o método *AddBitmap* do contentor que, recebe como parâmetro a nova imagem a adicionar, aplica o escalamento para a dimensão pré-definida de 100x100 *pixeis* e adiciona-a à lista de imagens. Esta dimensão pré-definida a aplicar às imagens faz com que cada imagem contenha 10000 *pixeis*. A execução desta operação é apresentada na Listagem 3.

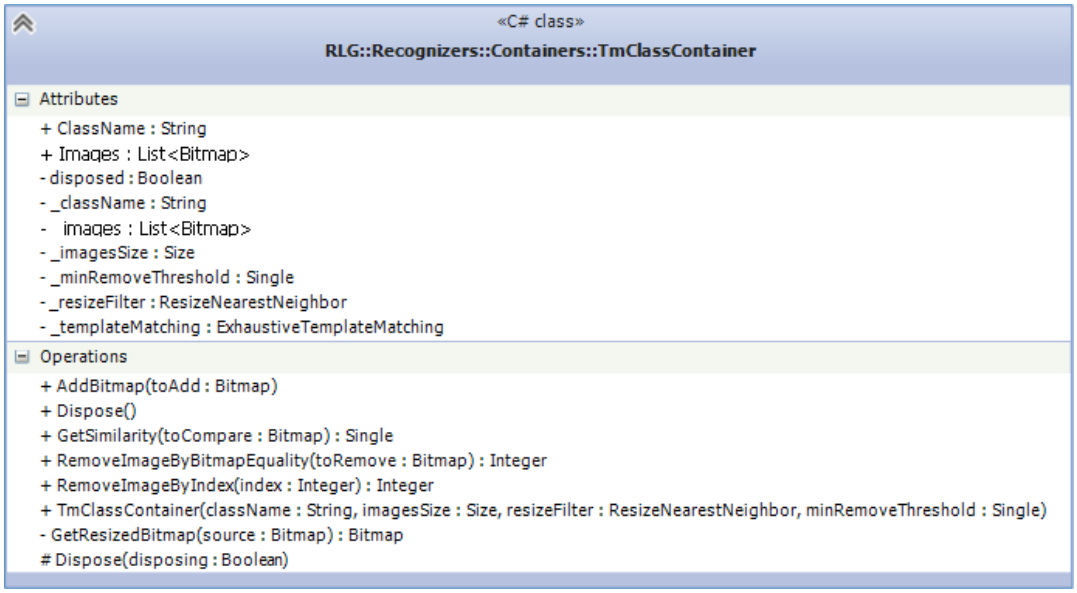


Figura 17 - UML da classe TmClassContainer

A classe *TemplateMatchingRecognizer* disponibiliza o método *GetBestClassSimilarity* que recebe como parâmetro uma imagem a classificar, invoca o método *GetSimilarity* de cada contentor de classe passando-lhes a imagem a classificar e por último retorna o nome e o valor de semelhança da classe que mais se assemelhe com a imagem comparada. No método *GetSimilarity* dos contentores, é aplicado o escalamento para a dimensão pré-definida e de seguida é utilizado o algoritmo de comparação exaustiva, presente na biblioteca AForge.Imaging, para se obter um valor de semelhança entre a imagem a comparar e cada uma das imagens de exemplo da classe. Após realizada a comparação com todas as imagens de exemplo é retornado o maior valor encontrado.

Para além de ser possível obter o maior valor de semelhança pelo processo acima descrito, foi implementado um mecanismo de reconhecimento temporal para aumentar a probabilidade de acerto no reconhecimento em tempo real. Neste mecanismo, dada uma nova imagem a classificar é obtido o maior valor de semelhança através do método *GetBestClassSimilarity* e adicionado o resultado a uma lista que contém as n últimas classificações. A classificação final é obtida com base nas n últimas classificações e é devolvida a classe que mais vezes se encontre nesta lista, minimizando o erro de uma imagem ser mal classificada quando é realizado reconhecimento em tempo real.

Na instanciação da classe *TemplateMatchingRecognizer* é recebida a altura e largura pré-definida a utilizar, um valor mínimo de semelhança para que seja possível atribuir a uma imagem o resultado de não identificação (classe de rejeição) e o tamanho da janela de reconhecimento temporal.

```
/// <summary>
/// Add template image with a given class name to container
/// </summary>
/// <param name="className">class name of the template</param>
/// <param name="templateImage">bitmap image</param>
/// <returns>Operation result state</returns>
public bool AddTemplate(string className, Bitmap templateImage)
{
    if (className == String.Empty || templateImage == null) return
false;
    var classContainer = _templatesClassContainer.Where(c =>
c.ClassName.Equals(className.Trim())).FirstOrDefault();
    if (classContainer == null)
    {
        var container = new TmClassContainer(className, _imagesSize,
_resizeFilter, _minRemoveBitmapTrashhold);
        container.AddBitmap(templateImage);
        _templatesClassContainer.Add(container);
        templateImage.Dispose();
    }
    else
        classContainer.AddBitmap(templateImage);
    return true;
}

/// <summary>
/// Add a template image to the images container.
/// The input Bitmap will be resized for the class images size;
/// </summary>
/// <param name="toAdd">Image to add</param>
public void AddBitmap(Bitmap toAdd) {
    if (toAdd == null) return;
    if (toAdd.Width != _imagesSize.Width || toAdd.Height !=
_imagesSize.Height)
    {
        var newBitmap = GetResizedBitmap(toAdd);
        if (newBitmap != null) _images.Add(newBitmap);
    }
    else
        _images.Add(toAdd);
}
```

Listagem 3 - Adicionar um novo padrão de exemplo

5.3.2. Análise de Componentes Principais

Como apresentado na secção 3.1.2, o PCA é uma técnica para encontrar padrões na informação e expressá-la de forma a destacar a suas semelhanças e diferenças. As características fornecidas a este algoritmo são as imagens da mão já processadas pelo algoritmo apresentado na secção 5.2. Uma vez que estas imagens poderão conter dimensões diferentes, o algoritmo de PCA aplica um escalamento para uma dimensão pré-definida a todas as imagens que processa. Como características das observações a fornecer ao algoritmo de PCA, foi decido utilizar o nível de intensidade dos pixéis das imagens. No entanto, como as imagens da mão já processadas pelo algoritmo apresentado na secção 5.2 estão no formato binário, este nível apenas pode ser ou zero (preto) ou um (branco). Assim, a cada nova imagem fornecida a este algoritmo é aplicado um escalamento para uma dimensão pré-definida e de seguida é extraído um vector contendo o nível de cada *pixel*. Foi ainda definido um contentor destes vectores, denominado *_imagesRChanelPixelsArray*, para guardar todas as características de exemplo. Para a implementação deste algoritmo foi desenvolvida a classe *PcaRecognizer* cuja representação UML é apresentada na Figura 18.

Para adicionar uma nova imagem de exemplo, esta classe disponibiliza o método *AddTemplate* que recebe como parâmetros o Bitmap da imagem da mão a adicionar e a respectiva classe. Depois de redimensionada a imagem e gerado o vector correspondente ao nível de cada *pixel*, o novo vector é adicionado ao contentor de características e o nome da classe é adiciona à lista de nomes de classe. Após criada a matriz de características é necessário normalizar as dimensões subtraindo a média de todas as imagens a cada uma delas, calcular a matriz de covariância, calcular os vectores próprios e projectar os vectores de características no espaço de componentes principais. Estes cálculos são realizado com recurso à classe *PrincipalComponentAnalysis* da biblioteca Accord.Statistics.Analysis que facilita todo este processo. Através da instrução *Compute* é possível obter a matriz de covariância e a matriz de vectores próprios já cálculados. Por último, para projectar os vectores de características no espaço de componentes principais é executado o método *Transform* sendo obtida a matriz de projecção.

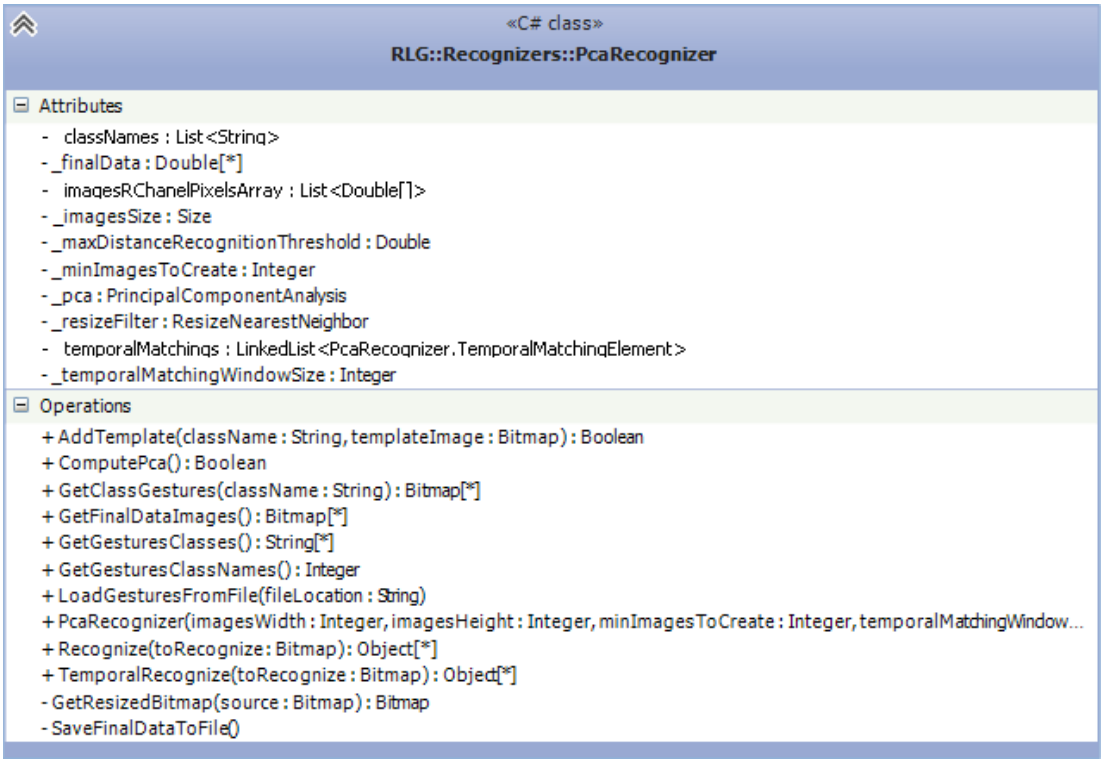


Figura 18 - UML da classe PcaRecognizer

Para classificar uma nova imagem é disponibilizado o método *Recognize* que recebe como parâmetro a imagem a classificar. Esta imagem é redimensionada para o tamanho predefinido nesta classe, é gerado o *array* correspondente à componente de cor vermelha de cada *pixel* e de seguida projectado no espaço de componentes principais. Por último, para se obter um valor de semelhança entre o vector de projecção obtido e todos os constituintes da matriz de projecção de exemplo, basta calcular a distância vectorial entre este novo vector e cada linha da matriz de projecção de exemplo. Para calcular esta distância é utilizada a distância de Manhattan dada por $d(p,q) = \sum_{i=1}^{n} \|p_i - q_i\|$. A menor distância encontrada representa a maior semelhança. Para que seja possível rejeitar uma classificação, a menor distância encontrada é ainda comparada com valor máximo de distância. Caso a distância encontrada seja inferior a este valor máximo então o gesto é classificado com sucesso sendo retornado o nome da classe e o valor da distância obtido. A execução desta operação é apresentada na Listagem 4.

```
/// <summary>
/// Recognize new image
/// </summary>
/// <param name="toRecognize"></param>
/// <returns>["class Name", min distance] or
["__UNKNOWN"]</returns>
public object[] Recognize(Bitmap toRecognize) {
    if(toRecognize == null) return new object[]{"__UNKNOWN null
source"};
    Bitmap toUse = (toRecognize.Width != _imagesSize.Width ||
toRecognize.Height != _imagesSize.Height) ?
GetResizedBitmap(toRecognize) : toRecognize;
    if (toUse == null) return new object[]{"__UNKNOWN resizing
source problem."};
    try
    {
        //Get pixel color array
        var pixelArray =
RLG.Helpers.Converters.Imaging.ExtractRChanelPixelsArray(toUse);
        var data = _pca.Transform(pixelArray);
        toUse.Dispose();
        toRecognize.Dispose();
        var minDistanceIndex = -1;
        var minDistanceFound = Double.MaxValue;
        for (int i = 0; i < _finalData.Count(); i++) {
            var currentDistance = 0.0d;
            for(int j= 0; j < _finalData[i].Count() -1; j++){
                currentDistance += Math.Abs(data[j] -
_finalData[i][j]);
            }
            if (currentDistance < minDistanceFound)
            {
                minDistanceFound = currentDistance;
                minDistanceIndex = i;
            }
        }

        if (minDistanceFound < _maxDistanceRecognitionTrashhold) {
            return new object[] { _classNames[minDistanceIndex],
minDistanceFound };
        }
        return new object[]{"__UNKNOWN"};
    }
    catch (Exception e)
    {
        toUse.Dispose();
        return new object[]{"Error" + e.Message};
    }
}
```

Listagem 4 - Processo de classificação PCA

Tal como no algoritmo de template matching, neste algoritmo, para além da possibilidade de obter a classe com menor distância a uma nova imagem, foi definido um mecanismo de reconhecimento temporal para aumentar a probabilidade de acerto

no reconhecimento em tempo real. Neste mecanismo, dada uma nova imagem a classificar é obtida a menor distância através do método *Recognize* e adicionado o resultado da classificação a uma lista que contém as n últimas classificações. Por fim, a classificação final é obtida com base nesta lista e devolvida a classe que mais vezes se encontre nas n últimas classificações. Desta forma, na classificação em tempo real se uma imagem for mal classificada este efeito é reduzido por este mecanismo.

5.4. Classificação e reconhecimento de gestos dinâmicos

Para a classificação e reconhecimento de gestos dinâmicos foram implementados dois algoritmos, alinhamento temporal dinâmico e modelos de Markov não-observáveis. Os dois algoritmos são adequados para reconhecimento de séries temporais, minimizando os efeitos de deslocamento. Os dois algoritmos recebem como características as coordenadas x, y e z das articulações das duas mãos e dos dois cotovelos. Para normalizar as coordenadas destas articulações, o referencial é convertido do sistema de coordenadas do dispositivo Kinect para o ponto intermédio da distância entre os ombros, ou seja, as coordenadas passam a ser medidas em relação a este novo ponto. As quatro articulações podem ser representadas pelo conjunto $A = \{a_1, a_2, a_3, a_4\}$ onde cada elemento a_i contém uma coordenada x, uma coordenada y e uma coordenada z. Para modelar este conjunto num só elemento, foi criado um vector onde cada articulação ocupa três posições, sendo a primeira referente à coordenada x, a segunda referente à coordenada y e a terceira referente à coordenada z, ou seja, obtém-se o conjunto $A' = \{a_{1x}, a_{1y}, a_{1z}, a_{2x}, a_{2y}, a_{2z}, a_{3x}, a_{3y}, a_{3z}, a_{4x}, a_{4y}, a_{4z}\}$. Desta forma, para modelar uma sequência temporal de observações é criada uma lista de elementos A' com n elementos, sendo n o número de observações.

5.4.1. Alinhamento Temporal Dinâmico

O algoritmo de alinhamento temporal dinâmico, como referido na secção 3.1.3, permite alinhar as duas sequências no tempo através de mapeamento não-linear sendo possível

medir semelhanças entre duas séries temporais. A implementação deste algoritmo foi baseada no projecto de código aberto "Kinect SDK Dynamic Time Warping (DTW) Gesture Recognition" [55]. Este projecto foi desenvolvido utilizando o Microsoft SDK para a aquisição da informação e permite a comparação de sequências temporais através de uma implementação do algoritmo DTW, onde para o alinhamento das sequências é utilizado o algoritmo do vizinho mais próximo. Como este projecto é baseado no Microsoft SDK, foi necessário modificar alguns métodos para que passem a suportar a informação das articulações provenientes do NiTE a 3 dimensões. Dado que no projecto original apenas era possível ter uma sequência de exemplo por classe, foi necessário alterar o método de inserção de uma nova sequência para que passe a suportar várias sequências de exemplo da mesma classe. Para a concretização deste algoritmo foi implementada a classe *DtwRecognizer* cuja representação UML é apresentada na Figura 19.

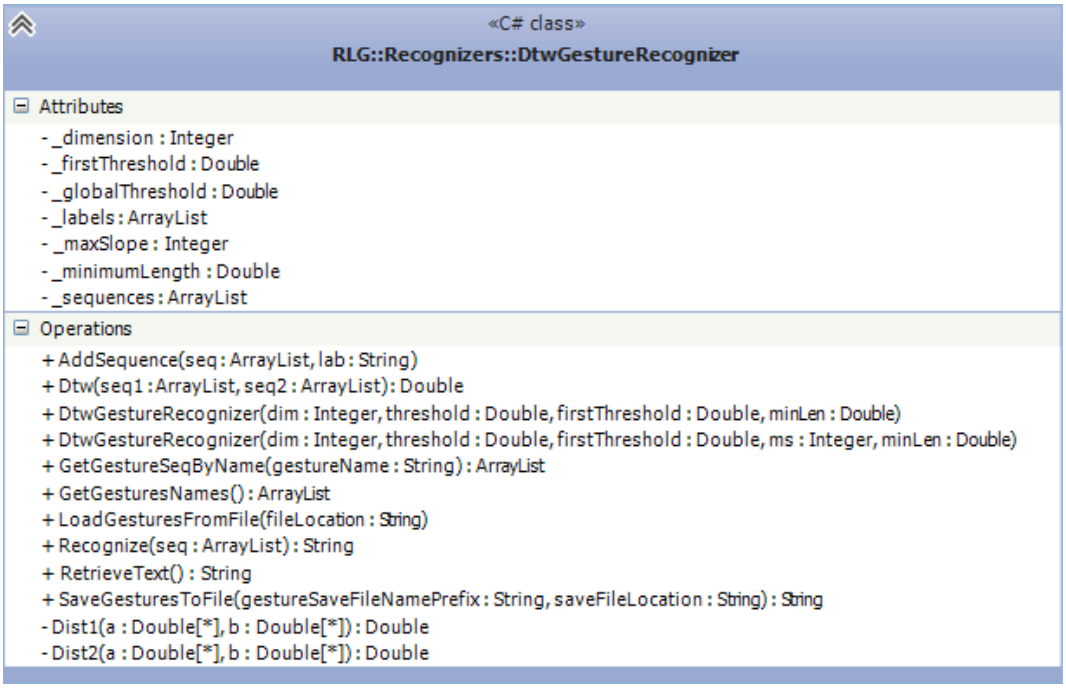


Figura 19 - UML da classe DtwGestureRecognizer

Para agrupar as sequências de observações apresentadas na secção 5.4 como o conjunto A', conjunto de coordenadas das articulações, esta classe contém um contentor

genérico ArrayList denominado *_sequences*. Para adicionar uma nova sequência de observações ao contentor, esta classe disponibiliza o método *AddSequence* que recebe por parâmetro a sequência a adicionar e o respectivo nome da classe, adiciona a nova sequência ao contentor *_sequences* e o nome da classe ao contentor de nomes da classe. A relação entre a sequência e a classe a que pertence é mantida pela mesma posição que têm nos dois contentores.

Para classificar uma nova sequência, esta classe disponibiliza o método *Recognize* que recebe como parâmetro a sequência a classificar. Para diminuir o número de processamentos desnecessários ao comparar sequências muito diferentes, inicialmente é calculada a distância entre a última observação da sequência a classificar e a última observação da sequência de exemplo (terminações). Caso esta distância não exceda um limite, então é iniciado o alinhamento entre as duas sequências para obter a menor distância entre a sequência de exemplo e todas as possíveis terminações da sequência a classificar. Depois de encontrada a menor distância entre a sequência a classificar e as sequências de exemplo que cumpram a condição inicial, é verificada se a distância total não excede um limite e caso isso não aconteça, então o gesto é reconhecido.

A esta classe foi também adicionado um método para exportar as sequências de exemplo para ficheiro e um método para carregar as sequências de exemplo a partir de um ficheiro.

5.4.2. Modelo de Markov Não-Observável

Como apresentado na secção 3.1.4, os modelos de Markov são adequados para modelar comportamentos temporais, assumindo que esses comportamentos contêm um número finito de estados internos, permitindo descobrir uma sequência adequada de estados para suportar as observações. Para a concretização deste algoritmo foi implementada a classe HmmRecognizer cuja representação UML é apresentada na Figura 20.

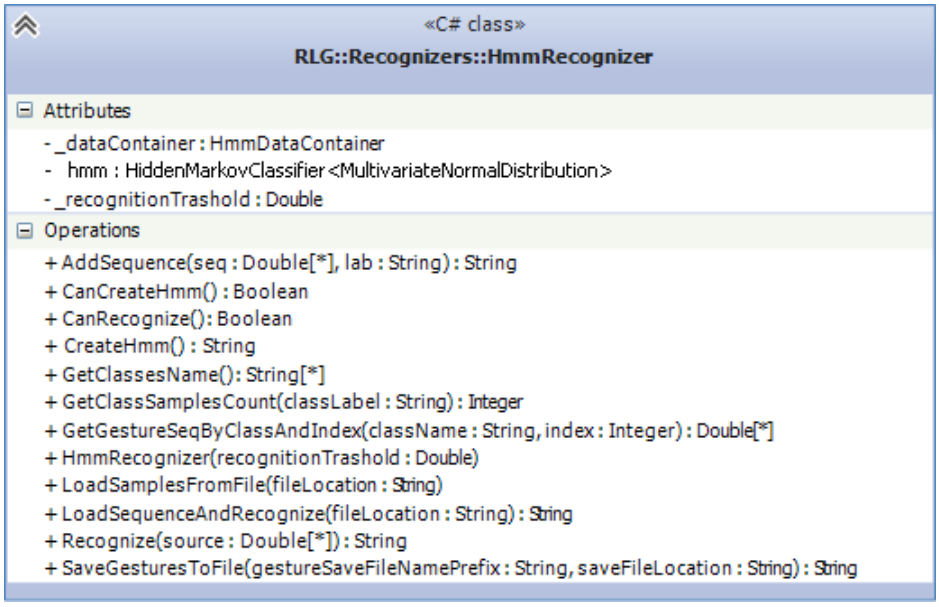


Figura 20 - UML da classe HmmRecognizer

Para guardar as sequências de observações pré-processadas, esta classe utiliza um objecto do tipo *HmmDataContainer*. Este contentor contém uma lista denominada de Samples com todas as sequências pré-processadas e os respectivos nomes de classe. As sequências que esta classe guarda são modeladas em objectos do tipo *HmmSequence* que contém uma lista de elementos A', apresentada na secção 5.4, e uma referência para o índice da lista de nomes de classe do contentor. As representações UML das classes *HmmDataContainer* e *HmmSequence* são apresentadas na Figura 21 e Figura 22, respectivamente.

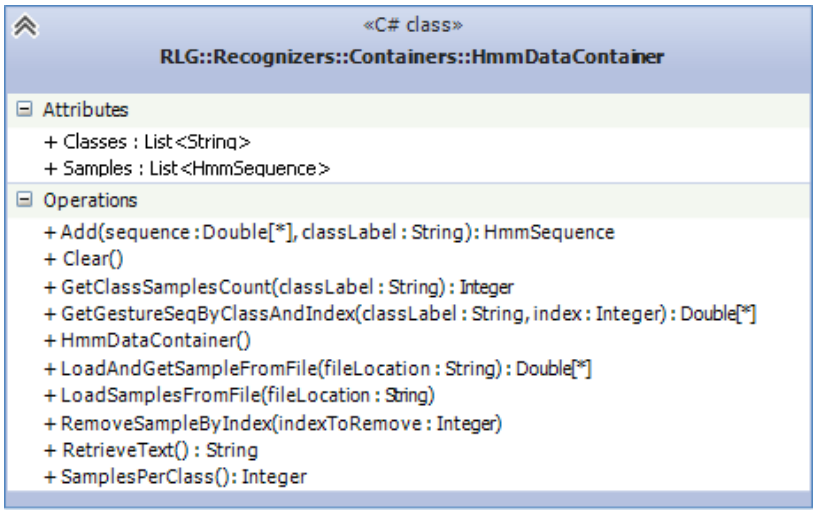


Figura 21 - UML da classe HmmDataContainer

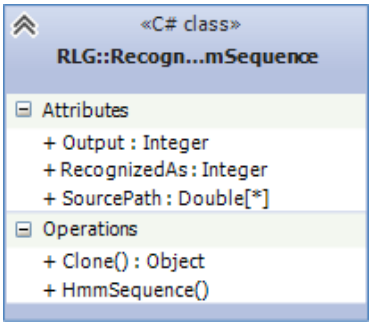


Figura 22 - UML da classe HmmSequence

Para adicionar uma nova sequência de exemplo, a classe *HmmRecognizer* disponibiliza o método *AddSequence* que recebe a nova sequência a adicionar e o respectivo nome da classe. De seguida este método passa os parâmetros recebidos ao método *Add* do contentor que verifica se o nome da classe ainda não se encontra na lista de nomes de classe e, caso não se encontre, é adicionada uma nova entrada e guardado o índice de inserção. Por último, é criado um novo objecto do tipo *HmmSequence* com a sequência e o índice guardado do nome da classe sendo de seguida adicionado à lista Samples. Foi ainda definido que para criar os HMMs são necessárias pelo menos três classes com três sequências cada. A classe *HmmRecognizer* disponibiliza o método *CreateHmm* para o efeito. Para criar o classificador esta classe recorre à biblioteca Accord.Statistics.Models.Markov para criar o objecto *HiddenMarkonClassifier*. Os HMMs a cada observação contínua com precisão *double* tem associada uma função de densidade de probabilidade. Como as características a fornecer a este algoritmo assumem valores contínuos, para a instanciação desta classe foi definida uma distribuição normal multivariada. Foi ainda definida uma topologia Ergódica onde qualquer estado pode transitar para qualquer outro estado. De seguida foi utilizado o algoritmo de treino para o classificador com recurso à classe HiddenMarkonClassifierLearning onde foi definido para utilizar o algoritmo de Baum-Welch com vista a estimar os parâmetros do modelo que melhor descrevem o processo. Depois de definido o processo de treino basta executar o método Run do algoritmo, passando como parâmetros a matriz contendo todas as observações e o *array* de nomes de classe. A partir deste ponto o modelo está criado e treinado sendo então possível classificar uma nova sequência.

Quando se pretende classificar uma nova sequência, a classe *HmmRecognizer* disponibiliza o método *Recognize* que recebe a sequência a classificar e executa o método *Compute* do classificador para obter a classe que produz a sequência de observações com maior probabilidade.

5.5. Reconhecimento contínuo

Um dos objectivos deste trabalho é que o sistema seja capaz de realizar o reconhecimento de gestos continuamente, ou seja, sem necessidade de indicar onde começa e acaba um novo gesto. No que toca a reconhecimentos de gestos estáticos esta funcionalidade é relativamente fácil de implementar sendo apenas necessário que, cada imagem da mão extraída pelo algoritmo apresentado na secção 5.2 seja passada ao método de reconhecimento que estiver a ser utilizado para o classificar. Relativamente ao reconhecimento de gestos dinâmicos e contínuos, foi necessário adoptar um mecanismo para dar suporte a este requisito.

Cada vez que o controlador devolve um novo conjunto de articulações, estas são convertidas para o formato do conjunto A', apresentado na secção 5.4, e de seguida adicionadas ao final de um *buffer* com um limite de 64 observações. Quando este *buffer* contém o mínimo de observações necessárias para se começar a realizar o reconhecimento, este conjunto de observações é passado ao método de reconhecimento que estiver a ser utilizado no momento para o reconhecer. Caso o gesto seja reconhecido, é apresentada a classificação atribuída à sequência e despejado o *buffer*. Caso o gesto não seja reconhecido e o *buffer* se encontre totalmente preenchido é removida a primeira observação.

5.6. Criação da Base de dados

Pelo facto de não existir nenhuma base de dados de gestos da LGP disponível e para que seja possível validar da solução proposta foi necessário construir uma base de dados de gestos estáticos e dinâmicos da Língua Gestual Portuguesa. Para se conseguir obter uma

maior qualidade de execução dos gestos, estes foram executados pela professora Patrícia Pessoa (ver Figura 23), a leccionar Língua Gestual Portuguesa na Associação Portuguesa de Surdos [56]. No total foram gravados 30 gestos estáticos executados com a mão direita e 28 gestos dinâmicos executados com ambas as mãos.

Figura 23 – Execução e gravação de gesto da LGP

5.6.1. Gestos estáticos

O conjunto de gestos estáticos da Língua Portuguesa é bastante reduzido e pouco utlizado em relação ao conjunto dos gestos dinâmicos. Os gestos estáticos são utilizados na maior parte para representar as letras do alfabeto e os números. Apenas uma limitada quantidade de gestos estáticos não se incluem nestas duas categorias. Das 26 letras do alfabeto, as letras D, K, Q, W, Y e Z requerem movimento para a sua execução e por esta razão não são considerados gestos estáticos sobrando 20 letras. Dos números de 0 a 9 apenas foi possível gravar 8 destes números. A forma da mão do gesto referente ao número 0 e à letra O não varia em nada e portanto estes dois gestos foram agrupados num só, denominado O/Zero. A postura da mão durante a execução do gesto referente ao número 9 impossibilitou que o processador *UserTracker* presente no NiTE fosse capaz

de seguir a mão e portanto impossibilitou a aquisição do mesmo. A este conjunto de gestos estáticos foram ainda adicionados os gestos Telefone e Mau, por corresponderem a gestos estáticos.

Para a gravação destes gestos, o sensor foi posicionado a 0,80m do nível do chão (aproximadamente) e o executor permaneceu sentado numa cadeira a 1,5 metros da câmara (aproximadamente). Após iniciada a detecção da mão, o executor realizou os gestos um a um variando a posição da mão sendo obtidas várias imagens de cada gesto perfazendo um total de 713 imagens. O número de imagens obtidas para cada gesto é apresentado na Tabela 1.

Gesto	Total	Gesto	Total
A	28	S	24
B	20	T	31
C	15	U	25
E	21	V	27
F	22	X	41
G	19	Telefone	19
H	26	Mau	24
I	24	1	26
J	24	2	20
L	23	3	21
M	23	4	20
N	23	5	24
O/Zero	30	6	17
P	28	7	22
R	26	8	20

Tabela 1 - Quantidade de imagens obtidas por classe para gestos estáticos

5.6.2. Gestos Dinâmicos

O conjunto de gestos dinâmicos presente na Língua Gestual Portuguesa é muito vasto do qual foram escolhidos 28 gestos. Como o controlador apresentado na secção 5.2 devolve as coordenadas x, y, z e o nome de cada articulação processada, foi adicionado um método ao controlador para permitir a modelação e gravação de todas as

características provenientes do *UserTracker*. Desta forma é possível guardar informação como a orientação das articulações e um valor quantitativo da confiança nos valores. Ao realizar a gravação de gestos dinâmicos desta forma, é possível posteriormente um possível aperfeiçoamento na classificação através da selecção destas características para fornecer a cada classificador. Para a gravação destes gestos, o sensor foi posicionado a sensivelmente a 0,80m do nível do chão (aproximadamente) e o executor permaneceu em pé a sensivelmente 2 metros da câmara (aproximadamente). Após iniciada a detecção do esqueleto, o executor realizou os gestos um a um sendo obtidas várias sequências de observações para cada gesto. Depois de revistas todas as sequências, removidos alguns dos erros e removidas as observações iniciais e finais referentes a transições entre gestos, foram seleccionadas 411 sequências. O número de sequências obtidas para cada gesto é apresentada na Tabela 2.

Gesto	Total	Gesto	Total
Acontece	14	Janela	12
Alto	17	Mosca	17
Ambulância	14	Muçulmano	11
Animal	13	Mulher	14
Aparelho Auditivo	14	Padre	15
Árvore	16	Praia	15
Casa	12	Sabonete	15
Céu	17	Sagitário	17
Ciúmes	10	Saia	15
Comer	19	Saudade	14
Copo	15	Segunda-feira	15
Cor	14	Terça-feira	14
Gostar	14	Vacina	17
Hospital	17	Visual	14

Tabela 2 - Quantidade de imagens obtidas por classe para gestos dinâmicos

(A) Pós-processamento de gestos

Depois dos gestos adquiridos foi necessário rever um a um para seleccionar os que foram correctamente gravados e remover os que continham erros ou estavam totalmente inutilizáveis. Para a revisão foi criada uma aplicação onde é possível carregar os ficheiros exportados aquando da gravação, visualizar cada observação das

sequências, aplicar diversas operações sobre as observações e por último exportar o resultado. Como operações implementadas destacam-se, a remoção de observações, a estabilização de articulações ou a redução do número total de observações de uma sequência. A interface de utilização desta aplicação é apresentada na Figura 24. Relativamente à remoção de observações de uma sequência é possível, remover a observação de apresentação actual, remover da observação inicial até à observação de apresentação actual e remover da observação actual até à última observação. Estas funções foram adicionadas pelo facto de existirem sequências onde um grande número de observações podem ser desprezadas, tanto no início como no fim da sequência, por não fazerem parte do gesto mas sim consideradas observações de transição entre gestos que podem ser apagadas. Outra das razões da implementação destas funções foi a necessidade de aproximar as coordenadas da observação final das sequências da mesma classe, isto porque, o algoritmo baseado em alinhamento temporal dinâmico desenvolvido tem como premissa o alinhamento dos gestos que terminam na mesma zona.

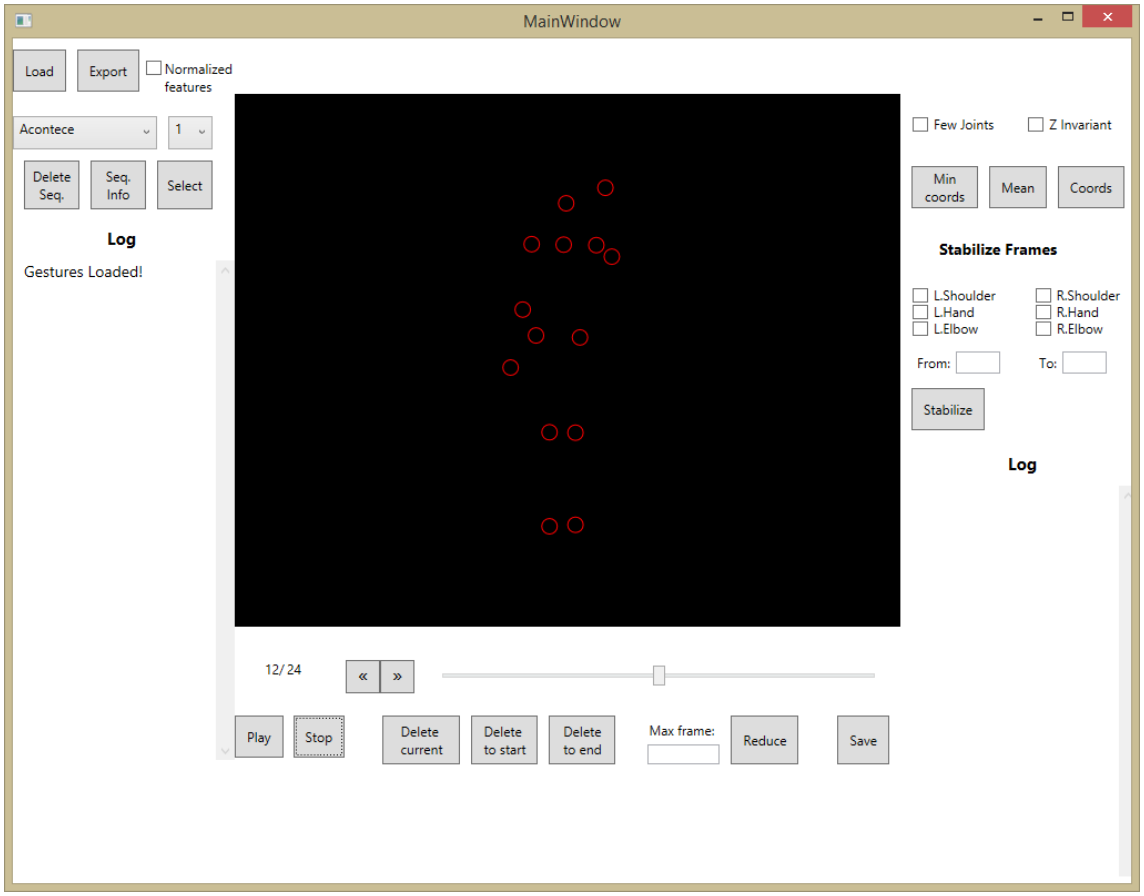


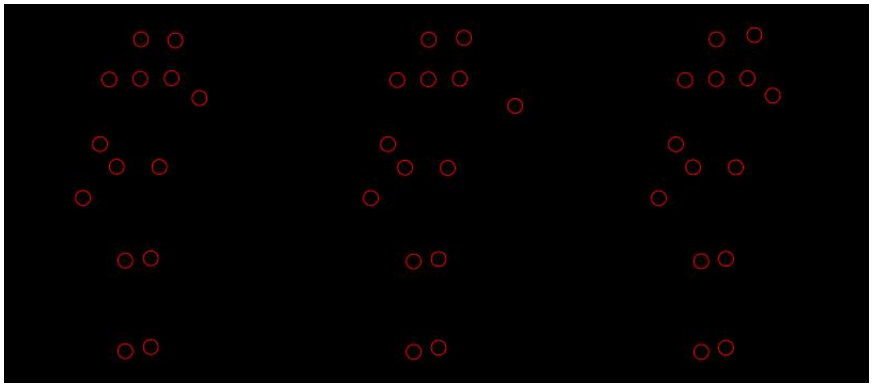
Figura 24 - Interface de utilização da aplicação de edição de sequências.

Durante a visualização das sequências gravadas foi observado que algumas delas apresentavam uma ou duas observações com pequenos erros na detecção de articulações. Na tentativa de computacionalmente remover estes erros foi implementada a função de estabilização de articulações. Nesta função, dada uma observação inicial, uma observação final e quais as articulações a estabilizar, o algoritmo calcula o posicionamento das articulações a estabilizar na observação inicial e final aplicando uma interpolação linear a cada coordenada (x, y, z) das observações intermédias. O código deste método é apresentado na Listagem 5.

```
private void
StabelizeJoint(RawGestureEditor.Helpers.Common.JointType jointType,
               ref RawJoint[][] joints)
{
    if (jointType == null || joints == null || joints.Count() < 3)
      return;

    var firstJointPosition = joints[0].Where(j =>
                  j.Type == jointType).First().raw_coord;
    var lastJointPosition = joints[joints.Count() - 1].Where(j =>
                  j.Type == jointType).First().raw_coord;

    var xFrameStep = (lastJointPosition.X - firstJointPosition.X) /
(joints.Count() - 1);
    var yFrameStep = (lastJointPosition.Y - firstJointPosition.Y) /
(joints.Count() - 1);
    var zFrameStep = (lastJointPosition.Z - firstJointPosition.Z) /
(joints.Count() - 1);

    for (int i = 1; i <= joints.Count() - 2; i++) {
        var newX = joints[i - 1].Where(j =>
            j.Type == jointType).First().raw_coord.X + xFrameStep;
        var newY = joints[i - 1].Where(j =>
            j.Type == jointType).First().raw_coord.Y + yFrameStep;
        var newZ = joints[i - 1].Where(j =>
            j.Type == jointType).First().raw_coord.Z + zFrameStep;
        joints[i].Where(j =>
            j.Type == jointType).First().raw_coord =
                   new System.Windows.Media.Media3D.Point3D(newX,
                                                        newY, newZ);
    }
}
```

Listagem 5 - Processo de estabilização de articulações do editor de sequências

Como exemplo da aplicação desta operação num cenário real, são apresentadas três observações na Figura 25 de uma sequência do gesto “Acontece” que contém um pequeno erro no cotovelo direito na observação número 30 (b).



(a) (b) (c)

Figura 25 - Estabilização de articulação com erro de um gesto "Acontece."

(a) imagem 29; (b) imagem 30; (c) imagem 31;

Na aplicação foi indicado que a estabilização a realizar será entre a observação 29 e 31 e que se pretende estabilizar o cotovelo direito, como apresentado na Figura 26.

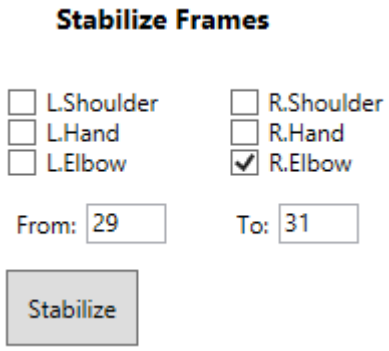
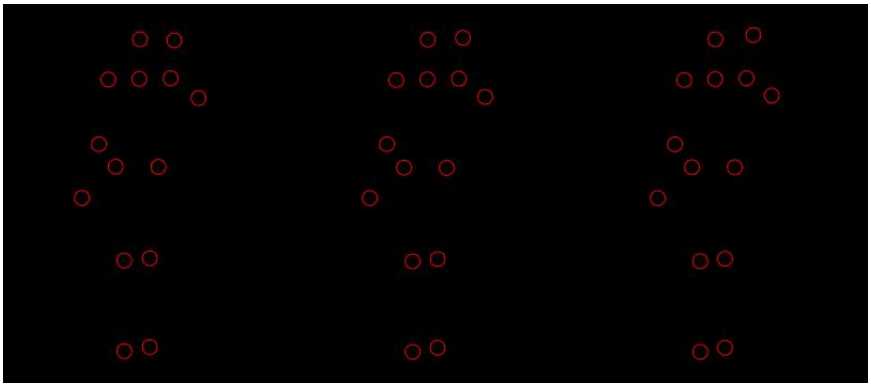


Figura 26 - Menu de selecção de opções de estabilização

Ao pressionar o botão "Stabilize", o cotovelo direito na observação número 30 (b) irá ser afectada com as coordenadas intermédias (x, y, z) das observações 29 e 31. O resultado desta operação é apresentado na Figura 27.



(a) (b) (c)

Figura 27 - Resultado da estabilização de articulação com erro de um gesto "Acontece."

(a) imagem 29; (b) imagem 30; (c) imagem 31)

Por último, foi implementada uma operação para uniformizar o número de observações de uma dada sequência. Nesta operação são calculadas quantas observações é necessário remover e o espaçamento entre remoções para que seja uniforme. No entanto, esta operação não chegou a ser utilizada.

6. Resultados

Neste capítulo são descritos os testes realizados a cada um dos algoritmos propostos e analisados os resultados obtidos. Para além da descrição de cada teste são apresentadas as observações seleccionadas para formarem os conjuntos de treino e de teste. Estas observações fazem parte da base de dados de gestos da LGP, apresentada na secção 5.6, construída para este trabalho. A primeira secção é referente aos testes realizados aos algoritmos de reconhecimento de gestos estáticos, *template matching* e análise de componentes principais. Na segunda secção são apresentados os testes aos algoritmos de reconhecimento de gestos dinâmicos, alinhamento temporal dinâmico e modelos de Markov não-observáveis.

6.1. Gestos estáticos

Para o reconhecimento de gestos estáticos da LGP, como apresentado na secção 5.3, foram implementados dois algoritmos para classificar com base nas observações da mão. Os testes realizados aos dois algoritmos são baseados em testes de validação cruzada onde são definidos 2 conjuntos, o primeiro é disponibilizado ao algoritmo em questão para o treino e o segundo para o teste. Os dois conjuntos são disjuntos entre si, ou seja, nenhuma observação do conjunto de treino está contida no conjunto de teste e vice-versa. As observações utilizadas para formarem os conjuntos foram retiradas da base de dados de gestos estáticos construída para este trabalho e apresentada na secção 5.6.1. Durante os testes foi registado o número de reconhecimentos correctos e errados por gesto e apresentados os resultados numa matriz de confusão que permite identificar a probabilidade de acerto dos algoritmos propostos na classificação de gestos estáticos da LGP.

6.1.1. Teste ao algoritmo *Template Matching*

A base de dados de 30 gestos estáticos, apresentada na secção 5.6.1, contém 713 observações da mão direita. A partir destas observações foram definidos 2 conjuntos disjuntos, o de treino e o de teste. Para o conjunto de treino foram seleccionadas 383 destas observações e para formar o conjunto de teste foram seleccionadas as restantes 330 observações. A distribuição do número de observações por classe de cada conjunto é apresentada na Tabela 3.

.

Gesto	Treino	Teste	Gesto	Treino	Teste
A	15	13	S	12	12
B	11	9	T	16	15
C	9	6	U	14	11
E	12	9	V	15	12
F	13	9	X	21	20
G	10	9	Telefone	11	8
H	13	13	Mau	12	12
I	13	11	1	14	12
J	12	12	2	11	9
L	12	11	3	12	9
M	12	11	4	11	9
N	12	11	5	12	12
O/Zero	17	13	6	9	8
P	15	13	7	12	10
R	14	12	8	11	9

Tabela 3 - Distribuição das observações por classe dos conjuntos de treino e de teste para o algoritmo TM

As observações presentes nos conjuntos e fornecidas a este algoritmo são directamente as imagens da mão processadas pelo algoritmo apresentado na secção 5.2.1. Depois de treinado o algoritmo com o conjunto de treino foi-lhe fornecido o conjunto de testes para se obter um valor quantitativo da probabilidade de acerto deste algoritmo no reconhecimento de gestos estáticos. A matriz de confusão apresentada na Tabela 4 contém o resultado das 330 classificações.

	A	B	C	E	F	G	H	I	J	L	M	N	O/Z	P	R	S	T	U	V	X	Tel	Mau	1	2	3	4	5	6	7	8
A	13																													
B		9																												
C			5												1															
E				9																										
F					9																									
G						9																								
H							13																							
I								11																						
J									12																					
L										11																				
M											11																			
N												11																		
O/Zero						2							11																	
P														13																
R															12															
S																12														
T																	15													
U																		11												
V																			12											
X																				20										
Tel																					8									
Mau																						12								
1																							12							
2																								9						
3																									9					
4																										9				
5																											12			
6																												8		
7																													10	
8																														9

Tabela 4 - Matriz de confusão de teste do algoritmo TM

De matriz de confusão gerada a partir dos resultados da classificação do conjunto de teste consegue-se inferir que, o algoritmo *template* matching num conjunto de 330 observações acertou na classificação de 327, errando apenas em 3. Estes três erros devem-se à forma da mão dos gestos 0/Zero e G serem muito semelhantes. O mesmo acontece para as letras C e R. Pela divisão do número de observações reconhecidas correctamente pelo total de observações classificadas obtém-se uma percentagem média de validações correctas de 99,09%. O algoritmo apresentou 1 erro para a letra C e dois erros para a letra O ou número 0 (O/Zero). No entanto, pelo facto do número de observações de teste da letra C ser inferior ao número de observações de teste do gesto O/Zero, a percentagem individual de classificações correctas foi influenciada. A letra C obteve uma percentagem individual de classificações correctas de 83,33% e o gesto O/Zero obteve uma percentagem individual de classificações correctas de 84,62%. O erro médio global que o algoritmo apresentou foi de 0,91%. Relativamente à performance, este algoritmo apresentou uma média de tempo de classificação de uma nova observação de 14,69ms.

6.1.2. Teste ao algoritmo Análise de Componentes Principais

Este algoritmo tem a vantagem relativamente ao anterior de reduzir o número de dimensões de cada observação conseguindo extrair apenas as componentes que melhor caracterizam cada observação. Como apresentado no capítulo 5.3.2 foi decido que as características a fornecer a este algoritmo correspondem ao valor de cada *pixel* das imagens binárias. Este algoritmo tem a particularidade de, o número de componentes principais calculadas a partir das observações de treino corresponderem ao número de observações do conjunto de treino, ou seja, quanto maior for o conjunto de treino inicial maior será o número de componentes principais calculadas. Para que seja possível realizar testes com diferentes números de componentes principais foram definidos 3 conjuntos disjuntos. Os dois primeiros serão utilizados para treino e o terceiro para teste. Os dois conjuntos de treino serão utilizados em duas fases diferentes de treino. Inicialmente é fornecido o primeiro conjunto de treino ao algoritmo para calcular as componentes principais e de seguida projectar esse mesmo conjunto sobre as componentes principais calculadas, gerando o conjunto inicial de observações aprendidas. O segundo conjunto de treino é de seguida fornecido ao algoritmo para que as suas observações sejam também projectadas sobre as componentes principais calculadas anteriormente e adicionado o resultado da projecção ao conjunto de observações aprendidas. Depois deste passo, o treino fica concluído e o conjunto de observações aprendidas contém a projecção de todas as observações de treino sobre as componentes principais calculadas a partir do primeiro conjunto de treino. Para se perceber as vantagens e desvantagens de serem utilizadas mais ou menos componentes principais, a cardinalidade do primeiro conjunto de treino foi variada e realizados testes distintos para cada variação. Tal como para o algoritmo *template matching*, para este algoritmo foram seleccionadas 383 observações para o conjunto total de treino e 330 observações para formar o conjunto de teste, apresentados na Tabela 3. Das 383 observações seleccionadas para o treino foram realizadas duas combinações de selecções que resultam em duas hipóteses de testes. A primeira contém 5 observações para o conjunto inicial de treino e as restantes fazem parte do segundo conjunto de treino. Para a segunda divisão foram seleccionadas 9 observações para o conjunto inicial de treino e as restantes fazem parte do segundo conjunto de treino. O teste onde o

conjunto inicial de treino é formado por 5 observações foi denominado, por nós, como "teste 5" e o teste onde o conjunto inicial de treino é formado por 9 observações foi denominado, por nós, como "teste 9". A cada imagem processada por este algoritmo é previamente aplicado um escalamento para uma dimensão pré-definida para que todas tenham a mesma dimensão. Para se perceber a vantagem de utilizar imagens com diferentes dimensões, o "teste 9" foi ainda divido em "teste 9 - 1" onde são utilizadas imagens com a dimensão de 50x50 *pixeis* e em "teste 9 - 2", onde são utilizadas imagens com a dimensão de 100x100 *pixeis*.

(A) Teste 5

Para este teste, o conjunto total de treino com 383 observações foi divido em dois subconjuntos. O conjunto inicial de treino contém 5 observações de cada gesto, perfazendo um total de 150, e o segundo conjunto de treino contém as restantes 233 observações. A distribuição do número de observações por classe de cada conjunto é apresentada na Tabela 5.

Gesto	1ª Treino	2º Treino	Teste	Gesto	1º Treino	2º Treino	Teste
A	5	10	13	S	5	7	12
B	5	6	9	T	5	11	15
C	5	4	6	U	5	9	11
E	5	7	9	V	5	10	12
F	5	8	9	X	5	16	20
G	5	5	9	Telefone	5	6	8
H	5	8	13	Mau	5	7	12
I	5	8	11	1	5	9	12
J	5	7	12	2	5	6	9
L	5	7	11	3	5	7	9
M	5	7	11	4	5	6	9
N	5	7	11	5	5	7	12
O/Zero	5	12	13	6	5	4	8
P	5	10	13	7	5	7	10
R	5	9	12	8	5	6	9

Tabela 5 - Distribuição das observações por classe dos conjuntos de treino e de teste para o algoritmo PCA para o "teste 5"

Para este teste foi definido que a dimensão das imagens a trabalhar será de 50x50 *pixeis* o que perfaz um total de 2500 *pixeis* por imagem, ou seja, cada observação é representada como um vector de 2500 posições contendo o valor de cada *pixel*. Após fornecido ao algoritmo o primeiro conjunto de treino é obtida uma matriz de 150x150 correspondeste às projecções das observações sobre as componentes principais calculadas. Neste ponto cada imagem passa a ser descrita por um vector de 150 projecções. De seguida são fornecidas as observações do segundo conjunto de treino que serão também projectadas sobre as componentes principais calculadas e adicionado o resultado à matriz de projecções, ficando esta com a dimensão final de 383x150. Por último foi fornecido o conjunto de testes para que cada observação seja classificada e registado o número de classificações correctas e erradas por gesto. O resultado destas classificações é apresentado na Tabela 6 como uma matriz de confusão.

	A	B	C	E	F	G	H	I	J	L	M	N	O	P	R	S	T	U	V	X	Tel	Mau	1	2	3	4	5	6	7	8
A	13																													
B		9																												
C			4			1									1															
E				9																										
F					9																									
G						9																								
H					2		10		1																					
I								11																						
J					1				10									1												
L										11																				
M											11																			
N												11																		
O													13																	
P														13																
R															12															
S				1										1		9									1					
T																	15													
U																		11		1										
V																			12											
X																				20										
Tel																					8									
Mau																						12								
1																							11						1	
2																								9						
3				1			1																		7					
4																										9				
5																											12			
6						1																						7		
7																							1						10	
8								2																						7

Tabela 6 - Matriz de confusão do "teste 5" ao algoritmo PCA

De matriz de confusão gerada a partir dos resultados do conjunto de teste consegue-se inferir que o algoritmo de análise de componentes principais, num conjunto de 330 observações acertou na classificação de 312 observações, errando em 18. A partir destes valores é possível calcular a percentagem média de classificações correctas de 94,55%. Pela divisão dos 18 erros pelas 330 observações classificadas obtém-se um erro global de 5,45%. Relativamente à performance, este algoritmo apresentou uma média de tempo de classificação de uma nova observação de 11,36ms.

(B) Teste 9 – 1

Neste teste pretende-se inferir se o aumento do número de observações do conjunto de treino inicial e consequente aumento do número de componentes principais, se traduz num aumento da percentagem total de classificações correctas. Este teste segue os mesmos princípios que o teste "teste 5", no entanto, para o primeiro conjunto de treino foi seleccionado o maior número de observações possível. Uma vez que a letra C e o número 6 só contêm 9 observações de treino, o máximo de observações conseguidas para o primeiro conjunto de treino foi de 9 observações. Foram portanto seleccionadas 9 observações de cada gesto para o primeiro conjunto de treino totalizando 270 observações e consequentemente o segundo conjunto de treino contém as restantes 113 observações do conjunto total. As distribuições das observações por classe de cada conjunto são apresentadas na Tabela 7.

Gesto	1ª Treino	2º Treino	Teste	Gesto	1º Treino	2º Treino	Teste
A	9	6	13	S	9	3	12
B	9	2	9	T	9	7	15
C	9	0	6	U	9	5	11
E	9	3	9	V	9	6	12
F	9	4	9	X	9	12	20
G	9	1	9	Telefone	9	2	8
H	9	4	13	Mau	9	3	12
I	9	4	11	1	9	5	12
J	9	3	12	2	9	2	9
L	9	3	11	3	9	3	9
M	9	3	11	4	9	2	9
N	9	3	11	5	9	3	12
O/Zero	9	8	13	6	9	0	8
P	9	6	13	7	9	3	10
R	9	5	12	8	9	2	9

Tabela 7 - Distribuição das observações por classe dos conjuntos de treino e de teste para o algoritmo PCA para o "teste 9 - 1"

Para este teste foi definido que a dimensão das imagens a trabalhar será também de 50x50 *pixeis* o que perfaz um total de 2500 *pixeis* por imagem, ou seja, a cada nova imagem fornecida a este algoritmo é aplicado um escalamento para uma dimensão de 50x50 *pixeis* e de seguida é extraído um vector de 2500 posições contendo o valor de cada *pixel*. Após fornecido ao algoritmo o primeiro conjunto de treino é obtida uma matriz de 270x270 correspondente às projecções das observações sobre as componentes principais calculadas. Neste ponto, cada imagem passa a ser descrita por um vector de 270 projecções em vez das 150 utilizadas no “teste 5”. De seguida, foram fornecidas as observações do segundo conjunto de treino que serão também projectadas sobre as componentes principais calculadas e o resultado adicionado à matriz de projecções, passando esta a ficar com a dimensão final de 383x270. Por último, foi fornecido o conjunto de teste ao algoritmo e registado o resultado das classificações na matriz de confusão apresentada na Tabela 8.

	A	B	C	E	F	G	H	I	J	L	M	N	O/Z	P	R	S	T	U	V	X	Tel	Mau	1	2	3	4	5	6	7	8
A	13																													
B		9																												
C			4			1									1															
E				9																										
F				1	8																									
G						8							1																	
H					1		11		1																					
I								11																						
J					1				11																					
L										11																				
M											11																			
N												11																		
O/Zero													13																	
P														13																
R															12															
S				1												10									1					
T																	15													
U																		10		1										
V																			12											
X																				19					1					
Tel																					8									
Mau																		1				11								
1																							12							
2																								9						
3							1																		8					
4																										9				
5																											12			
6						2																						6		
7																													10	
8																														9

Tabela 8 - Matriz de confusão do "teste 9 - 1" ao algoritmo PCA

De matriz de confusão gerada a partir dos resultados da classificação de todo o conjunto de teste consegue-se inferir que, o algoritmo análise de componentes principais para imagens de 50x50 *pixeis* e com 270 projecções por observação classificou correctamente 315 observações num total de 330 classificações. Através destes valores consegue-se calcular que o algoritmo apresentou uma percentagem média de classificações correctas de 95,45%. Este valor representa um aumento de acerto de 0,9% em relação ao “teste 5”. Através da divisão dos 15 erros pelas 330 observações classificadas obtém-se um erro total de 4,55%. Relativamente à performance, este algoritmo apresentou uma média de tempo de classificação de uma nova observação de 20,08ms que, apesar de ser ligeiramente superior ao valor obtido no teste anterior, continua a ser admissível para reconhecimento em tempo real.

(C) Teste 9 - 2

Neste teste pretende-se inferir se o aumento da dimensão das imagens para 100x100 *pixeis* se traduz num aumento da percentagem total de classificações correctas. Para tal, este teste segue os mesmos princípios que o teste "teste 9 - 1", onde para os conjuntos de treino e de teste foram também escolhidas as mesmas observações, apresentadas na Tabela 7. Foram seleccionadas 9 observações de cada gestos para o primeiro conjunto de treino totalizando 270 observações o que fez com que o segundo conjunto de treino contenha as restantes 113 observações do conjunto total. A diferença entre este teste e o teste anterior está na dimensão pré-definida das imagens a trabalhar. No teste anterior foi definida uma dimensão de 50x50 *pixeis* o que perfaz um total de 2500 *pixeis* por imagem e neste teste foi definida que a dimensão passa a ser de 100x100 *pixeis* o que perfaz um total de 10000 *pixeis* por imagem. Como o número de observações do primeiro conjunto de treino não se alterou, após a primeira fase de teste é também obtida uma matriz de 270x270 projecções. Neste ponto cada imagem passa a ser descrita por um vector de 270 projecções sobre as componentes principais calculadas. De seguida, foram fornecidas as observações do segundo conjunto de treino que serão também projectadas sobre as componentes principais e adicionado o resultado à matriz de projecções, passando a ficar com a dimensão de 383x270. Por último, foi fornecido o conjunto de teste para que este algoritmo classifique cada observação. O resultado é apresentado na Tabela 9 como uma matriz contendo as classificações correctas e erradas por gesto.

	A	B	C	E	F	G	H	I	J	L	M	N	O/Z	P	R	S	T	U	V	X	Tel	Mau	1	2	3	4	5	6	7	8
A	13																													
B		9																												
C			3			1							1		1															
E				6																					3					
F					9																									
G						9																								
H					1		12																							
I					1			10																						
J					1				11																					
L										11																				
M											11																			
N												11																		
O/Zero						1							12																	
P														13																
R															12															
S				1		1										10														
T																	15													
U																		11												
V																			12											
X																				20										
Tel																					8									
Mau																						12								
1																							12							
2																								9						
3							1																		8					
4																										9				
5																											12			
6																												8		
7																													10	
8																														9

Tabela 9 - Matriz de confusão do "teste 9 - 2" ao algoritmo PCA

Desta matriz de confusão consegue-se inferir que, o algoritmo análise de componentes principais para imagens de 100x100 *pixeis* e com 270 projecções por observação classificou correctamente 317 observações num total de 330 classificações. Relativamente ao “teste 9-1 “, neste teste foram removidos dois erros pelo aumento da dimensão das imagens sendo obtida uma percentagem média de classificações correctas de 96,06% que representa um aumento de 0,61% em relação ao teste anterior. Relativamente ao erro foi obtido um erro total de classificação de 3,94%. Este algoritmo apresentou uma média de 67,42ms para classificar uma nova imagem, que ainda é praticável para classificação em tempo real.

6.2. Gestos Dinâmicos

Para o reconhecimento de gestos dinâmicos da LGP, como apresentado na secção 5.4, foram implementados dois algoritmos para classificar sequências de observações de coordenadas a 3 dimensões das duas mãos e dos cotovelos. Os testes realizados aos dois algoritmos são baseados em testes de validação cruzada onde são definidos 2 conjuntos, o primeiro é disponibilizado ao algoritmo em questão para o treino e o segundo é utilizado para teste. Estes conjuntos sãos disjuntos entre si, ou seja, nenhuma observação do conjunto de treino está contida no conjunto de teste e vice-versa. As observações utilizadas para formarem os conjuntos foram retiradas da base de dados de 28 gestos dinâmicos criada apresentada na secção 5.6.2. Para o conjunto de treino foram seleccionadas 213 sequências de observações e para formar o conjunto de teste foram seleccionadas as restantes 198 sequências. Nos testes realizados aos dois algoritmos de classificação de gestos dinâmicos, foram utilizados os mesmos conjuntos de treino e de teste cuja distribuição do número de sequências por gesto de cada conjunto é apresentada na Tabela 10.

Gesto	Treino	Teste	Gesto	Treino	Teste
Acontece	7	7	Janela	6	6
Alto	9	8	Mosca	9	8
Ambulância	7	7	Muçulmano	6	5
Animal	7	6	Mulher	7	7
Aparelho Auditivo	7	7	Padre	8	7
Árvore	8	8	Praia	8	7
Casa	6	6	Sabonete	8	7
Céu	9	8	Sagitário	9	8
Ciúmes	5	5	Saia	8	7
Comer	10	9	Saudade	7	7
Copo	8	7	Segunda-feira	8	7
Cor	7	7	Terça-feira	7	7
Gostar	7	7	Vacina	9	8
Hospital	9	8	Visual	7	7

Tabela 10 - Distribuição das sequências de observações por classe dos conjuntos de treino e de teste para os algoritmos de classificação de gestos dinâmicos

Durante os testes foi registado o número de reconhecimentos correctos e errados por gesto e apresentados os resultados numa matriz de confusão que permite identificar o grau de acerto dos algoritmos propostos na classificação de gestos estáticos da LGP.

6.2.1. Teste ao algoritmo Alinhamento Temporal Dinâmico

Para o teste de validação cruzada, inicialmente foi fornecido ao algoritmo o conjunto de treino contendo 213 sequências de observações e posteriormente fornecidas as sequências de observações de teste para classificação, sendo projectado o resultado numa matriz de confusão apresentada na Tabela 11.

	Acon	Alto	Ambu	Anim	Apar. Aud.	Árvo	Casa	Céu	Ciúm	Comer	Copo	Cor	Gostar	Hosp.	Janela	Mosca	Muçu	Mulher	Padre	Praia	Sabon	Sagi	Saia	Sauda	Seg. F.	Ter. F.	Vaci	Visu
Acontece	4		3																									
Alto		8																										
Ambulância			6																					1				
Animal				5																				1				
Aparelho Auditivo				1	4																			2				
Árvore						8																						
Casa							6																					
Céu								8																				
Ciúmes									5																			
Comer										8		1																
Copo											7																	
Cor												7																
Gostar													7															
Hospital														8														
Janela															6													
Mosca																8												
Muçulmano																	5											
Mulher																		7										
Padre													3						4									
Praia																				7								
Sabonete																					7							
Sagitário																						8						
Saia																							7					
Saudade																								7				
Segunda Feira																									7			
Terça Feira																										7		
Vacina																											8	
Visual																									1			6

Tabela 11 - Matriz de confusão do teste ao algoritmo DTW

De matriz de confusão gerada a partir dos resultados da classificação de todo o conjunto de teste, consegue-se inferir que o algoritmo de alinhamento temporal dinâmico classificou correctamente 185 observações num total de 198 classificações, tendo errado em 13 classificações. Pela divisão do número de observações classificadas correctamente pelo total de classificações obtém-se uma percentagem média de classificações correctas de 93,43%. Através da divisão dos 13 erros pelas 198

observações classificadas obtém-se um erro total de 6,57%. Os erros observados referem-se a gestos que, apesar de terem diferentes durações, são executados nas mesmas zonas do corpo e apresentam uma grande proximidade na localização das mãos e cotovelos. Isto acontece porque o algoritmo de alinhamento temporal dinâmico tenta calcular a melhor correspondência entre duas sequências independentemente da sua duração. Relativamente à performance, este algoritmo apresentou uma média de tempo de classificação de uma nova observação de 117,37ms.

6.2.2. Teste aos modelos de Markov não-observáveis

Para este algoritmo foi também realizado um teste de validação cruzada com base nos dois conjuntos, de teste e de treino, apresentados na secção 6.2. Inicialmente foi fornecido ao algoritmo o conjunto de treino contendo 213 sequências de observações, de seguida foi criado o HMM e por último foram fornecidas as sequências de observações de teste para que o algoritmo as classifique. O resultado da classificação de cada gesto é apresentada na matriz de confusão presente na Tabela 12.

	Acon	Alto	Ambu	Anim	Apar. Aud.	Árvo	Casa	Céu	Ciúm	Comer	Copo	Cor	Gostar	Hosp.	Janela	Mosca	Muçu	Mulher	Padre	Praia	Sabon	Sagi	Saia	Sauda	Seg. F.	Ter. F.	Vaci	Visu	Unknown
Acontece	6		1																										
Alto		8																											
Ambulância			7																										
Animal				6																									
Aparelho Auditivo					7																								
Árvore						8																							
Casa							3																						3
Céu								8																					
Ciúmes									5																				
Comer										9																			
Copo											7																		
Cor												7																	
Gostar													7																
Hospital														8															
Janela															5														1
Mosca																8													
Muçulmano																	5												
Mulher																		7											
Padre																			7										
Praia																				7									
Sabonete																					7								
Sagitário																						8							
Saia																							7						
Saudade																								7					
Segunda Feira																									7				
Terça Feira																										7			
Vacina																											8		
Visual																												7	

Tabela 12 - Matriz de confusão do teste ao algoritmo HMM

Os resultados obtidos neste teste foram bastante bons tendo sido correctamente classificadas 193 das 198 sequências de observações. Pela divisão do número de sequências correctamente classificados pelo total de sequências obtém-se uma percentagem de reconhecimentos correctos de 97,47%, que representa um aumento de 4,04% em relação ao algoritmo de DTW. Mais uma vez a proximidade na localização da execução dos gestos "Acontece" e "Ambulância" provocou que ocorra 1 erro de classificação. Este erro também ocorreu no teste realizado ao algoritmo DTW embora que tenha confundido os dois gestos mais que uma vez. É possível também observar que 3 gestos "Casa" e um 1 "Janela" foram classificados como "desconhecido". Se assumirmos a classificação de "desconhecido" como um erro obtemos um total de 5 erros, totalizando 2,53% de erro global. Relativamente à performance, este algoritmo apresentou uma média de tempo de classificação de uma nova observação de 322,22ms, o que representa um aumento de 204,85ms no tempo necessário para a classificação, em relação ao algoritmo DTW.

7. Conclusões e trabalho futuro

7.1. Conclusões

O trabalho por nós realizado focou-se na implementação e teste de um sistema de reconhecimento de gestos estáticos e dinâmicos da LGP. Para a aquisição de estímulos do corpo humano foi utlizado o dispositivo sensorial Kinect. Devido ao facto de não existir nenhuma solução totalmente eficaz para se obter um maior detalhe da mão através da informação disponibilizada pelo Kinect, foi colocada a hipótese de utilizar o dispositivo sensorial Leap Motion cuja principal função é a de adquirir o detalhe das mãos e dos dedos. Após alguns testes, esta hipótese foi posta de parte porque o curto campo de visão deste dispositivo não é favorável para soluções em que as mãos navegam para além de 600mm de distância do sensor. Para além disso, a precisão apresentada por este sensor no que toca ao detalhe dos dedos ficou muito aquém do esperado, pois a configuração da mão e a perspectiva que o sensor tem dela, faz com que os dedos sejam perdidos em diversas situações. Numa segunda tentativa de se obter maior detalhe da mão, foi colocada a hipótese de recorrer à *framework* Kinect 3D Hand Tracking, que segundo a equipa que a criou, é capaz de estimar 26 graus de liberdade da mão a partir das imagens fornecidas pelo Kinect. No entanto, após alguns testes a esta *framework*, verificou-se que a inicialização da detecção da mão não é trivial para o utilizador final e mesmo que se consiga iniciar a detecção, a mão e os dedos são perdidos com grande facilidade, não se revelando uma solução viável para o problema. Posto isto, para a modelação e extracção de características do corpo do executor, foi utilizada somente a informação disponibilizada pelo Kinect.

Reconhecimento de gestos estáticos

Para o processamento e modelação da informação da mão, apresentamos um algoritmo que, a partir de uma imagem de profundidade e do centróide da mão, consegue extrair uma imagem de recorte contendo a mão centrada nela e consegue remover o fundo da imagem de recorte. A partir da utilização deste algoritmo para processar as imagens da base de dados de gestos estáticos criada para este trabalho, conseguiu-se inferir que o

algoritmo apresentado consegue com sucesso recortar a imagem da mão e extrair correctamente o fundo na maior parte das imagens. No entanto, na fase de remoção do fundo observou-se que o filtro de intensidade revelou não ser uma solução 100% eficaz devido ao facto dos limites a manter serem fixos e poder resultar na remoção de pequenas porções de informação da mão. Foi ainda testada a hipótese de criar um esqueleto da forma da mão através de um algoritmo de *thinning*, que no entanto, não apresentou ser uma solução viável para o reconhecimento de gestos em tempo real devido ao tempo necessário para calcular o esqueleto de cada imagem da mão. Para a classificação de gestos estáticos apresentamos dois algoritmos com os quais obtivemos uma percentagem de reconhecimentos correctos muito positiva. Com algoritmo *template matching* obtivemos 99,09% de reconhecimentos correctos ao classificar correctamente 327 observações num total de 330. Relativamente ao algoritmo baseado na distância vectorial dos vectores de projecção das características sobre as componentes principais, no melhor dos casos ("teste 9-2"), foi obtida uma percentagem de reconhecimentos correctos de 96,06%, ao reconhecer correctamente 317 de 330 observações. Este algoritmo demonstrou ainda minimizar o número de erros tanto pelo aumento de 5 para 9 componentes principais calculadas como pelo aumento da dimensão das imagens a processar de 50x50 *pixeis* para 100x100 *pixeis*. Estes dois algoritmos apresentaram ser aplicáveis para o reconhecimento em tempo real, sendo obtida uma média de tempo de classificação de uma nova observação de 14,69ms com o algoritmo *template matching* e de 67,42ms com o algoritmo análise de componentes principais.

Reconhecimento de gestos dinâmicos

Para o reconhecimento de gestos dinâmicos em tempo real, apresentamos dois algoritmos de classificação que apenas utilizam como características as sequências de coordenadas 3D das duas mãos e dos cotovelos.

Relativamente ao algoritmo DTW obtivemos uma percentagem de reconhecimentos correctos de 93,43% ao classificar correctamente 185 das 198 sequências de observações. Através da análise dos 13 erros obtidos conseguiu-se inferir que, os erros ocorreram devido á proximidade na localização de execução dos gestos porque o

algoritmo DTW tenta calcular a melhor correspondência entre duas sequências independentemente da sua duração. Quer isto dizer que, pelo facto de uma parte da sequência a classificar ter coordenadas muito próximas de uma parte de outra sequência de outra classe, leva a que o algoritmo classifique erradamente. Relativamente à performance, este algoritmo apresentou uma média de tempo de classificação de uma nova observação de 117,37ms.

Nos teste ao algoritmo HMM, obtivemos 97,47% de reconhecimentos correctos ao classificar correctamente 193 de 198 sequências de observações. Relativamente ao algoritmo anterior houve uma aumento de 4,04%, embora que, o tempo médio necessário para a classificação de uma sequência tenha aumentado 204,85ms. Este algoritmo apenas errou numa classificação ao confundir um gesto "Acontece" com um gesto "Ambulância", provavelmente pela mesma razão que apresentada para o algoritmo anterior. Outros 4 gestos foram ainda classificados como "desconhecido", totalizando 5 classificações incorrectas que se traduz em 2,53% de erro global.

7.2. Trabalho futuro

Perante a solução proposta e os testes realizados é possível identificar alguns temas que poderiam conduzir a melhorias do sistema. Em primeiro lugar, o conjunto de observação da base de dados devia ser alargado e para além de conter mais gestos, deveria conter observações de diferentes executores. Era também vantajoso se fossem desenvolvidos e testados outros algoritmo de classificação. Relativamente ao processamento das imagens da mão, gostaríamos de perceber como os classificadores de gestos estáticos responderiam ao adicionar invariância na rotação da mão. Uma melhoria possível ao reconhecimento em tempo real seria o ignorar de observações de transição entre gestos e o de ignorar a repetição de classificação de gestos, como acontece actualmente no reconhecimento contínuo de gestos estáticos. A conjugação de gestos estáticos e dinâmicos de forma a criar frases da Língua Portuguesa seria também um desafio interessante e visto como uma grande mais-valia. Esta conjugação associada a regras linguísticas permitiria retirar o contexto de uma frase e utiliza-lo como característica

desambiguadora nos algoritmos de classificação. A implementação de um avatar permitiria posteriormente complementar o sistema de reconhecimento com a tradução de Língua Portuguesa para Língua Gestual Portuguesa permitindo a comunicação bilateral. O mundo do 3D está em constante evolução e durante a realização deste trabalho surgiram novos dispositivos, tais como Creative Senz3D [57] ou Intel Real Sense [58], que vêm mais uma vez mudar o paradigma das HCI introduzindo a possibilidade de captar mais estímulos naturais do corpo humano e de forma mais precisa com um baixo custo. Seria interessante perceber se estes dipositivos permitiriam melhorar o reconhecimento de gestos da LGP.

8. Referências

[1] "Kinect For Windows," [Online]. Available: http://www.microsoft.com/en-us/kinectforwindows/develop/. [Acedido em 6 3 2014].

[2] J. L. Hernandez-Rebollar, R. W. Lindeman e N. Kyriakopoulos, "A Multi-Class Pattern Recognition System for Practical Finger Spelling Translation," em *Proceedings of the 4th IEEE International Conference on Multimodal Interfaces*, Washington, DC, USA, 2002.

[3] J.-L. Hernandez-Rebollar, "A New Instrumented Approach for Translating the American Sign Language into Sound and Text," The George Washington University, 2003.

[4] T. Starner, J. Weaver e A. Pentland, "A wearable computer based American sign language recognizer," em *Wearable Computers, 1997. Digest of Papers., First International Symposium on*, 1997.

[5] H. Sagawa, M. Takeuchi e M. Ohki, "Description and Recognition Methods for Sign Language Based on Gesture Components," em *Proceedings of the 2Nd International Conference on Intelligent User Interfaces*, New York, NY, USA, 1997.

[6] Z. Zafrulla, H. Brashear, T. Starner, H. Hamilton e P. Presti, "American Sign Language Recognition with the Kinect," em *Proceedings of the 13th International Conference on Multimodal Interfaces*, New York, NY, USA, 2011.

[7] Y. Li, "Hand gesture recognition using Kinect," em *Software Engineering and Service Science (ICSESS), 2012 IEEE 3rd International Conference on*, 2012.

[8] Z. Ren, J. Meng, J. Yuan e Z. Zhang, "Robust Hand Gesture Recognition with Kinect Sensor," em *Proceedings of the 19th ACM International Conference on Multimedia*, New York, NY, USA, 2011.

[9] U. Mahbub, H. Imtiaz, T. Roy, M. S. Rahman e M. A. R. Ahad, "A template matching approach of one-shot-learning gesture recognition," *Pattern Recognition Letters ,* vol. 34, nº 15, pp. 1780-1788, 2013.

[10] J. Dias, P. Nande, N. Barata e A. Correia, "OGRE - open gestures recognition engine," em *Computer Graphics and Image Processing, 2004. Proceedings. 17th Brazilian Symposium on*, 2004.

[11] R. N. de Almeida, "Portuguese Sign Language Recognition via Computer Vision and Depth Sensor," Tese Mestrado, ISCTE IUL, 2011.

[12] A. P. d. A. Sousa, "Interpretação da língua gestual portuguesa," Tese Mestrado, Universidade de Lisboa, 2012.

[13] "Kinect Sign Language Translator," Microsoft, [Online]. Available: http://blogs.msdn.com/b/msr_er/archive/2013/10/29/kinect-sign-language-translator.aspx. [Acedido em 27 5 2014].

[14] "MotionSavvy," [Online]. Available: http://www.motionsavvy.com/. [Acedido em 18 4 2014].

[15] I. Sim-Sim, I. Duarte e M. Ferraz, A Língua materna na educação básica: competências nucleares e níveis de desempenho, Ministério da Educação, 1997.

[16] S. Fischer, History of Language, Reaktion Books, 1999.

[17] C. Helena, M. Mariana, M. Marta e E. Paula, *Programa curricular de Língua Gestual Portuguesa,* Ministério da Educação, 2007.

[18] M. A. Amaral, A. Coutinho e M. R. D. Martins, Para uma gramática da língua gestual portuguesa, Caminho, 1994.

[19] A. Baltazar, Dicionário de língua gestual portuguesa, Porto Editora, 2010.

[20] I. H. Faria, L. Henriques e M. Martins, "Predicados de Movimento em Língua Gestual Portuguesa," *Polifonia,* vol. 4, pp. 87-98, 2001.

[21] M. R. Delgado-Martins, "Língua gestual: uma linguagem alternativa," em *Introdução à linguística geral e portuguesa*, I. H. Faria, E. R. Pedro, I. Duarte e C. Gouveia, Edits., Caminho, 1996, pp. 103-114.

[22] G. R. M. Turk, "Perceptual user interfaces," *Communications of the ACM,* vol. 43, nº 3, pp. 33-34, March 2000.

[23] D. J. Sturman e D. Zeltzer, "A Survey of Glove-based Input," *IEEE Comput. Graph. Appl.,* vol. 14, nº 1, pp. 30-39, #jan# 1994.

[24] E. Kaiser, A. Olwal, D. McGee, H. Benko, A. Corradini, X. Li, P. Cohen e S. Feiner, "Mutual Disambiguation of 3D Multimodal Interaction in Augmented and Virtual Reality," em *Proceedings of the 5th International Conference on Multimodal Interfaces*, New York, NY, USA, 2003.

[25] K. Khoshelham, "Accuracy analysis of kinect depth data," *ISPRS workshop laser scanning,* vol. 38, nº 5, p. W12, 2011.

[26] K. Khoshelham e S. O. Elberink, "Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications," *Sensors,* vol. 12, nº 2, pp. 1437-1454, 2012.

[27] C. Sun, T. Zhang, B.-K. Bao, C. Xu e T. Mei, "Discriminative exemplar coding for sign language recognition with Kinect.," *IEEE Trans Cybern.,* vol. 43, nº 1, pp. 1418-1428, October 2013.

[28] "Microsoft News Center," Microsoft, [Online]. Available: http://news.microsoft.com/2013/02/11/xbox-execs-talk-momentum-and-the-future-of-tv/. [Acedido em 18 1 2014].

[29] "Kinect Hardware Info," [Online]. Available: http://openkinect.org/wiki/Hardware_info. [Acedido em 4 2 2014].

[30] "OpenKinect," [Online]. Available: http://openkinect.org/wiki/Main_Page. [Acedido em 4 02 2014].

[31] "OpenNI," [Online]. Available: http://www.openni.org/. [Acedido em 6 3 2014].

[32] "Leap Motion Documentation," [Online]. Available: https://developer.leapmotion.com/documentation. [Acedido em 8 2 2014].

[33] W. Newman e R. Sproull, Principles of interactive computer graphics, McGraw-Hill, 1979.

[34] L. Smith, "A tutorial on Principal Components Analysis," 2002.

[35] L. Vladutu, "Non-rigid Shape Recognition for Sign Language Understanding," *WTOS,* vol. 8, nº 12, pp. 1263-1272, #dec# 2009.

[36] T.-N. Nguyen, H.-H. Huynh e J. Meunier, “Static Hand Gesture Recognition Using Principal Component Analysis Combined with Artificial Neural Network,” *Jounal of Automation and Control Engineering,* vol. 3, nº 1, pp. 40-45, February 2005.

[37] D. M. Capilla, *Sign Language Translator using Microsoft Kinect XBOX 360,* Girona, Spain, 2012.

[38] P. Vamplew, “Recognition of Sign Language Gestures using neural Networks,” em *Australian Journal of Intelligent Information Processing Systems*, 1998.

[39] C. Vogler e D. Metaxas, “ASL recognition based on a coupling between HMMs and 3D motion analysis,” em *Computer Vision, 1998. Sixth International Conference on*, 1998.

[40] T. Starner e A. Pentland, “Real-time American Sign Language recognition from video using hidden Markov models,” em *Computer Vision, 1995. Proceedings., International Symposium on*, 1995.

[41] J. Yang e Y. Xu, “Hidden Markov Model for Gesture Recognition,” Robotics Institute, Pittsburgh, PA, 1994.

[42] P. Blunsom, *Hidden Markov Models,* 2004.

[43] P. Senin, “Dynamic time warping algorithm review,” Department of Information and Computer Sciences, University of Hawaii, Honolulu, Hawaii, 2008.

[44] W. Stokoe, “Sign Language Structure: An Outline of the Visual Communication Systems of the American Deaf,” *Journal of Deaf Studies and Deaf Education,* vol. 19, nº 4, pp. 3-37, 1960.

[45] E. Costello, Random House Webster's Concise American Sign Language Dictionary, N.Y.: Random House, 1999.

[46] C. Vogler e D. Metaxas, “Toward Scalability in ASL Recognition: Breaking Down Signs into Phonemes,” em *Gesture-Based Communication in Human-Computer Interaction*, vol. 1739, A. Braffort, R. Gherbi, S. Gibet, D. Teil e J. Richardson, Edits., Springer Berlin Heidelberg, 1999, pp. 211-224.

[47] D. Terdiman, “Leap Motion: 3D hands-free motion control, unbound,” 2012.

[48] *Kinect 3D Hand Tracking framework.*

[49] S. Falahati, “NiWrapper.Net,” [Online]. Available: https://github.com/falahati/NiWrapper.Net. [Acedido em 16 4 2014].

[50] “AForge.Net,” [Online]. Available: http://www.aforgenet.com/. [Acedido em 2 7 2014].

[51] “Accord.Net,” [Online]. Available: https://code.google.com/p/accord/. [Acedido em 12 8 2014].

[52] Microsoft, “Windows Presentation Foundation,” [Online]. Available: http://msdn.microsoft.com/en-us/library/ms754130(v=vs.110).aspx. [Acedido em 16 2 2014].

[53] B. Ionescu, D. Coquin, P. Lambert e V. Buzuloiu, “Dynamic Hand Gesture Recognition Using the Skeleton of the Hand,” *EURASIP Journal on Advances in Signal Processing,* vol. 2005, nº 13, p. 236190, 2005.

[54] “Thinning,” [Online]. Available: http://homepages.inf.ed.ac.uk/rbf/HIPR2/thin.htm. [Acedido em 26 7 2014].

[55] “Kinect SDK Dynamic Time Warping (DTW) Gesture Recognition,” [Online]. Available: Kinect SDK Dynamic Time Warping (DTW) Gesture Recognition. [Acedido em 20 08 2014].

[56] “Associação Portuguesa de Surdos,” [Online]. Available: http://www.apsurdos.org.pt/. [Acedido em 5 4 2014].

[57] “Creative Senz3D,” [Online]. Available: http://us.creative.com/p/web-cameras/creative-senz3d. [Acedido em 22 9 2014].

[58] “Intel RealSense,” [Online]. Available: http://www.intel.com/content/www/us/en/architecture-and-technology/realsense-depth-technologies.html. [Acedido em 16 9 2014].

[59] M. Lamar, M. Bhuiyan e A. Iwata, “Hand alphabet recognition using morphological PCA and neural networks,” em *Neural Networks, 1999. IJCNN '99. International Joint Conference on*, 1999.

[60] B. Bauer, H. Hienz e K. F. Kraiss, “Video-based continuous sign language recognition using statistical methods,” em *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, 2000.