Instituto Superior de Engenharia do Porto

Interface Caligráfica de Escrita no Ar

Pedro Miguel Oliveira Leitão

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Sistemas Gráficos e Multimédia

Orientador: Doutor João Paulo Jorge Pereira

Co-orientador: Mestre António Abel Vieira de Castro

Júri:

Presidente: Doutora Maria de Fátima Coutinho Rodrigues

Vogais:

Doutor António Fernando Vasconcelos Cunha Castro Coelho

Doutor João Paulo Jorge Pereira

Mestre António Abel Vieira de Castro

Porto, Outubro 2012

Resumo

Na atualidade, está a emergir um novo paradigma de interação, designado por Natural User Interface (NUI) para reconhecimento de gestos produzidos com o corpo do utilizador. O dispositivo de interação Microsoft Kinect foi inicialmente concebido para controlo de videojogos, para a consola Xbox360. Este dispositivo demonstra ser uma aposta viável para explorar outras áreas, como a do apoio ao processo de ensino e de aprendizagem para crianças do ensino básico. O protótipo desenvolvido visa definir um modo de interação baseado no desenho de letras no ar, e realizar a interpretação dos símbolos desenhados, usando os reconhecedores de padrões Kernel Discriminant Analysis (KDA), Support Vector Machines (SVM) e $N. O desenvolvimento deste projeto baseou-se no estudo dos diferentes dispositivos NUI disponíveis no mercado, bibliotecas de desenvolvimento NUI para este tipo de dispositivos e algoritmos de reconhecimento de padrões. Com base nos dois elementos iniciais, foi possível obter uma visão mais concreta de qual o *hardware* e *software* disponíveis indicados à persecução do objetivo pretendido.

O reconhecimento de padrões constitui um tema bastante extenso e complexo, de modo que foi necessária a seleção de um conjunto limitado deste tipo de algoritmos, realizando os respetivos testes por forma a determinar qual o que melhor se adequava ao objetivo pretendido. Aplicando as mesmas condições aos três algoritmos de reconhecimento de padrões permitiu avaliar as suas capacidades e determinar o $N como o que apresentou maior eficácia no reconhecimento.

Por último, tentou-se averiguar a viabilidade do protótipo desenvolvido, tendo sido testado num universo de elementos de duas faixas etárias para determinar a capacidade de adaptação e aprendizagem destes dois grupos. Neste estudo, constatou-se um melhor desempenho inicial ao modo de interação do grupo de idade mais avançada. Contudo, o grupo mais jovem foi revelando uma evolutiva capacidade de adaptação a este modo de interação melhorando progressivamente os resultados.

Palavras-chave: Classificadores estatísticos, NUI, Kinect, Nuvens de pontos, Interação baseada em gestos, Calibração de dispositivos NUI, Interação natural, Interação caligráfica, Reconhecimento de padrões.

Abstract

A new interaction paradigm, designated by Natural User Interface (NUI), for the recognition of gestures produced with the user's body, is emerging. The interaction device Microsoft Kinect was originally designed to control video games, for console Xbox360. This device proves to be a viable wager to explore other areas such as support to the teaching and learning process for children of elementary school. The developed prototype aims to define a mode of interaction for drawing letters in the air, and perform the interpretation of the symbols by means of the pattern recognizers Kernel Discriminant Analysis (KDA), Support Vector Machines (SVM) and $ N. The development of this project was based on the study of different NUI devices available on the market, development libraries NUI for this type of devices and algorithms for pattern recognition. Based on the two initial elements, it was possible to get a more concrete vision of which of the available hardware and software adequate to achieve the intended goal.

The pattern recognition is a very extensive and complex subject, so that it was necessary to select a limited set of such algorithms and perform tests to determine which one was the best suited to the intended purpose. Applying the same conditions to the three pattern recognition algorithms, it allowed to evaluate their skills and determine the $ N as the highest efficiency in recognition.

Finally, the prototype's underlying ideas were tested in a universe of elements from two age groups in order to determine the ability of adaptation and learning of these two groups. In this study, it was found an initial better perform to mode of interaction of the older age group. However, the younger group was revealing an evolutionary adaptability to this mode of interaction improving results progressively.

Keywords: Statistical classifiers, NUI, Kinect, Clouds of points, Gesture based interaction, Calibration of NUI devices, Natural Interaction, Calligraphic Interaction, Patterns Recognition.

Agradecimentos

Esta dissertação não representa apenas o resultado de extensas horas de estudo, reflexão e trabalho durante as diversas etapas que a constituem. É igualmente o colmatar de um objetivo académico a que me propus, e que não seria possível sem a principal ajuda dos meus orientadores, Doutor João Paulo Jorge Pereira e o Mestre António Abel Vieira de Castro, pelos seus sábios conselhos, recomendações e contagioso entusiasmo ao longo de todo este ano letivo.

Índice

Lista de Figuras

Lista de Gráficos

Lista de Tabelas

Acrónimos e Símbolos

Lista de Acrónimos

API	Application Programming Interface
CL	Code Laboratories
CLR	Common Language Runtime
CMOS	Complementary Metal Oxide Semiconductor
CNMAT	Center for New Music and Audio Technology
FAAST	Flexible Action and Articulated Skeleton Toolkit
FOV	Field of View
GDIF	Gesture Description Interchange Format
GNU	GNU General Public License
GSS	Golden Section Search
HIG	Human Interface Guidelines
ICP	Iterative Closest Point
IR	Infra-vermelho
JPEG	Joint Photographic Experts Group
KDA	Kernel Discriminant Analysis
KPCA	Kernel Principal Component Analysis
LDA	Linear Discriminant Analysis
MIDI	Musical Instrument Digital Interface
MSIL	Microsoft Intermediate Language
NATO	North Atlantic Treaty Organization
NIME	New Interfaces for Musical Expression
NUI	Natural User Interface
OSC	Open Sound Control
PCA	Principal Component Analysis
PCL	Point Cloud Library
ROS	Robot Operating System
SDK	Software Development Kit
SMO	Sequential Minimal Optimization
SoC	System-on-a-Chip
SVM	Support Vector Machines
TUIO	Tangible User Interface Objects
VRPN	Virtual Reality Peripheral Network
XACT	Cross Plataform Audio Creation Tool
XML	Extensible Markup Language

1 Introdução

A interação com equipamentos eletrónicos usando teclado, rato ou outro controlador manual, tem vindo a ser bastante vulgarizada. O dispositivo Microsoft Kinect pretende colmatar as limitações humano-máquina existentes, através de uma interação baseada no reconhecimento de gestos realizados com o corpo. Este dispositivo, inicialmente desenvolvido para a utilização em videojogos, tem vindo a espoletar interesse junto do público e a demonstrar diversas potencialidades. A aplicabilidade do Kinect não se limita unicamente aos videojogos, existindo um vasto leque de áreas nas quais pode ser utilizado. O âmbito deste projeto visa a aplicação do Kinect, juntamente com o reconhecimento de padrões, à área da aprendizagem, integrando alguns algoritmos de forma a interpretar e identificar as letras correspondentes aos símbolos desenhados pelo utilizador, auxiliando assim o desenvolvimento motor e apoiando o processo de aprendizagem.

Nos tempos que correm, com a globalização e o crescimento tecnológico acelerado, as pessoas têm vindo a aderir cada vez mais a equipamentos digitais, em detrimento da escrita convencional (papel e lápis). Contudo, e apesar da sociedade se encontrar cada vez mais informatizada e dependente da tecnologia, a escrita convencional continua a ter um papel indispensável no quotidiano das pessoas, sobrepondo-se, em determinadas situações, à escrita digital. A escrita de textos, equações ou o desenho de esquemas nas salas de aulas constituem um bom exemplo da sua indispensabilidade. Estas tarefas tornam-se mais funcionais quando aliadas a uma interface tradicional.

Ao longo do tempo têm surgido diversas aplicações com este tipo de interfaces, refere Tomas Sylverberg [Sylverberg et al., 07]. Este autor propõe, como forma de apoio a militares, o uso de dispositivos móveis que lhes permitam desenhar símbolos NATO[1], essenciais na orientação

[1] Conjunto de símbolos militares padronizados, para marcações em mapas pela North Atlantic Treaty Organization (NATO).

dos soldados no terreno. O uso de dispositivos permitiria a partilha de informações do terreno, reportando a informação desenhada aos restantes elementos dispersos no campo.

1.1 Solução implementada

Para a solução desenvolvida, pensou-se num processo natural, com base em gestos simples, que permitisse às crianças desenharem caracteres no espaço, caracteres esses sugeridos aleatoriamente por uma aplicação informática dotada de uma interface natural. Os desenhos são posteriormente identificados e associados aos caracteres correspondentes. No final, são é determinado o número de identificações bem-sucedidas dos desenhos realizados.

1.2 Introdução do protótipo

Com base num estudo dos vários dispositivos NUI existentes no mercado e no conjunto de bibliotecas que permitem desenvolver *software* para esses equipamentos, foram escolhidos para integrar o protótipo, o dispositivo Microsoft Kinect e a biblioteca OpenNI/NITE. Têm a seu cargo a tarefa de detetar os gestos produzidos pelo utilizador. Posteriormente, estes gestos são interpretados pelo protótipo, espoletando as ações adequadas como movimentar o cursor, desenhar pontos ou selecionar eventos. O protótipo vai sugerindo aleatoriamente caracteres ao utilizador. Este desenha-os e um reconhecedor caligráfico, baseado no algoritmo $N [Anthony and Wobbrock, 2007a], trata de o identificar e associar ao respetivo carácter.

1.3 Contributos do trabalho

Esta dissertação, bem como o trabalho subjacente, propõe-se contribuir para a área educativa, desenvolvendo um protótipo focalizado no ensino de crianças do ensino básico. Considera-se a motivação dos alunos para a aprendizagem um aspeto fundamental para o sucesso da aprendizagem. Desta forma, o trabalho desenvolvido associa-se às novas tecnologias, usando um modo de intenção intuitivo e inovador, visando despertar neles o interesse pela aprendizagem.

1.4 Organização da dissertação

No capítulo 2, será introduzido o conceito de interação NUI, bem como o *hardware* disponível e a sua aplicação em consolas de videojogos.

O capítulo 3 é destinado à análise das características de funcionamento do dispositivo Microsoft Kinect.

No capítulo 4, serão apresentadas as várias bibliotecas de desenvolvimento NUI existentes e a plataforma aplicada no desenvolvimento do protótipo.

O capítulo 5 descreve os principais tipos de reconhecedores de padrões e aborda os algoritmos de reconhecimento de padrões analisados no âmbito desta dissertação.

O trabalho realizado e a respetiva avaliação são descritos nos capítulos 6 e 7, respetivamente.

Por último, no capítulo 8, procede-se à conclusão da dissertação e traçam-se as perspetivas de trabalho futuro.

2 Hardware NUI

Neste capítulo será introduzido o conceito de interação com dispositivos NUI e quais as valias que este recente paradigma de interação vem oferecer aos seus utilizadores. São também apresentados vários equipamentos NUI disponíveis no mercado, enumeradas as características que os constituem, o seu modo de funcionamento, e descritas as particularidades inerentes a cada um dos sistemas.

2.1 Natural User Interface

O conceito de interface NUI, contrariamente à maioria das interfaces de computadores, consiste numa interface baseada em gestos naturais produzidos com o corpo humano. Este tipo de interfaces distingue-se dos tradicionais comandos pela sua invisibilidade, e por implicarem uma aprendizagem rápida e simples por parte do utilizador [Wigdor and Wixon, 2011]. São várias as aplicações NUI desenvolvidas para a área da educação. Também a Microsoft Partners in Learning disponibiliza publicamente um conjunto de ferramentas para fomentar a criação deste tipo de aplicações inovadoras na área da educação. Uma das aplicações desse tipo é o 'KinectMath', desenvolvido pela Universidade de Washington Bothell, e que serve como utensílio para auxiliar a alunos na aprendizagem da matemática, contrariando o processo tradicional em que o professor expõe a matéria e os alunos tentam assimilá-la. Com isto foi possível tornar o processo menos maçador para o aluno, podendo agora interagir e manipular as funções por si, despertando-lhes maior interesse e tornando a aprendizagem menos teórica e mais prática [Kinect Education, 2012]. Outra aplicação é o 'KinectPaint' que, como o nome indica, é usado para realizar pinturas, à semelhança da versão tradicional Windows, e permite que os jovens desenhem e pintem na tela do monitor usando interação NUI [Kinect Paint, 2012].

2.2 Dispositivos usados em consolas de jogos

Nesta subsecção serão apresentados vários dispositivos de interação NUI, desenvolvidos para as consolas de videojogos atualmente disponíveis no mercado. Será descrito o funcionamento dos vários dispositivos assim como as particularidades inerentes na interação com cada um destes dispositivos com base na especificação da sua tecnologia.

2.2.1 Microsoft Kinect

O Kinect surgiu de uma conferência realizada em Junho de 2009, sendo que o seu lançamento para o mercado aconteceu apenas em Novembro de 2010, tendo vendido nos primeiros 60 dias 8 milhões de unidades, e alcançado no ano seguinte um volume de vendas de mais 18 milhões [Tscherrig, 2011].

O projeto que esteve na origem do Microsoft Kinect designou-se 'Projeto Natal', e veio revolucionar a área da visão computacional tendo como base a tecnologia PrimeSense. A tecnologia adotada baseou-se no sistema de luz padronizada, e no microcontrolador PS1080 SoC, concebendo assim um dispositivo de controlo de vídeojogos para a consola Xbox 360 da Microsoft. Este dispositivo visa oferecer ao utilizador uma interação totalmente inovadora, sem necessidade de toque ou equipamento extra, usando apenas a movimentação do corpo do utilizador e os gestos realizados para interagir com a aplicação e espoletar ações. Em comparação com soluções menos atuais, este equipamento destaca-se pela capacidade de obter dados em profundidade de um cenário, em tempo real, com um nível de detalhe razoável e não necessitar de quaisquer apetrechos dispendiosos, como luvas ou fatos equipados com sensores [Villaroman et al., 2011].

O lançamento oficial do Microsoft Kinect ocorreu nos Estados Unidos da América a 4 de Novembro de 2010. Desde cedo a comunidade esperava ansiosamente para a sua utilização para outros fins que não exclusivamente os vídeojogos. Após o seu lançamento, a empresa Adafruit, sediada em Nova Iorque, e que se dedica a descobrir vulnerabilidades no *hardware* de dispositivos [Adafruit, 2011], anunciou um desafio ao qual atribuía uma recompensa financeira de $1,000 à primeira pessoa que desenvolvesse um *driver* livre para o Kinect independente do sistema operativo. Dois dias após o lançamento deste desafio, AlexP anunciou ter conseguido atingir o objetivo, tendo publicado um vídeo onde o demonstrava. Contudo, no dia seguinte, 7 de Novembro, disse apenas colocar à disposição o *driver* criado mediante um contributo de $10,000 de forma a apoiar o seu trabalho. Assim, a 10 de Novembro, aquando do lançamento do Kinect na Europa, surge uma declaração de Hector Martin, natural de Espanha, que prova também ter conseguido descobrir as vulnerabilidades do dispositivo, explorando-as de modo a desenvolver o *driver*. Este, por sua vez, divulgou o seu trabalho, ganhando automaticamente a recompensa da Adafruit e contribuindo para o

aparecimento do *driver* libfreenect . O libfreenect, também designado de OpenKinect, consiste num driver *open-source*[2] desenvolvido de raiz para interação NUI [Rogers, 2011].

No mês seguinte surgiram dois novos lançamentos para o Kinect: um deles foi o OpenNI, que surgiu da associação de organizações com interesse em fomentar a introdução no mercado de aplicações NUI. Contrariamente ao libfreenect, não pretendia que fosse um driver NUI desenvolvido de raiz, mas uma arquitetura padronizada na qual dispositivos NUI se pudessem integrar. O outro lançamento aconteceu a 8 de Dezembro, e deu origem ao driver CL NUI Platform of Code Laboratories, que teve por base o código desenvolvido por AlexP [Villaroman et al., 2011].

O dispositivo Kinect tem vindo a ser aplicado e explorado em algumas das áreas de investigação como [Rogers, 2011]:

- Sistemas de visão robótica;
- Mapeamento do interior de salas;
- Controlo de luzes com gestos;
- Interface com o computador através de linguagem gestual;
- Música com gestos;
- Teclado virtual;
- Instrução (exemplo: Karaté, dança);
- Apresentação baseada em gestos.

Na constituição deste dispositivo (Figura 1) podemos encontrar um conjunto de microfones para aplicações que envolvam o reconhecimento de comandos por voz, uma câmara RGB com frequência de amostragem de 30 fotogramas por segundo e resolução de 640x480 píxeis. Além disto, inclui ainda outra câmara de infravermelhos (IR) com uma resolução de 640x480 píxeis de 30 fotogramas por segundo, usada para medir a profundidade dos elementos no cenário através da luz estruturada projetada sobre eles pelo emissor de IR e posteriormente refletida para a câmara referida.

[2] Licença de *software* livre, também conhecida por gratuita de código aberto, e que não restringe a venda ou distribuição do programa.

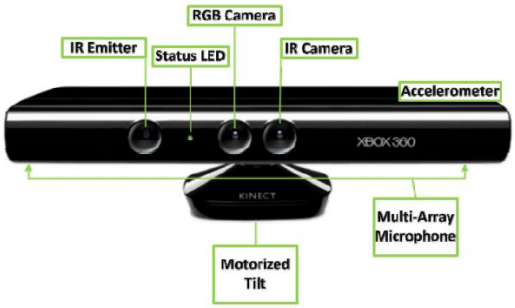


Figura 1 - Constituição do Microsoft Kinect [Mehlmann e tal., 2011].

No que se refere às limitações impostas às dimensões do cenário (Figura 2), este dispositivo especifica um limite de ação considerando que a profundidade aconselhável de uso é de 0.8 a 3.5 metros, sendo que a região que permite obter uma melhor precisão vai dos 1.2 aos 2.0 metros [Castaneda and Navab, 2011]. Além da distância ao sensor estar limitada, o dispositivo, antes de iniciar a captura de pontos, necessita de calibrar as duas imagens recolhidas pelas duas câmaras para dar início ao rastreio do cenário em busca de utilizadores. Pelo que este processo pode ser realizado manualmente. Contudo, poderia tornar-se bastante fastidioso, recorre geralmente a bibliotecas que incluem funções para o efeito, permitindo calibrar automaticamente o sensor. As duas câmaras apresentam diferentes campos de visão (FOV): a de profundidade possui 58° horizontais e 45° verticais; já a RGB tem um campo de visão ligeiramente superior. Assim, a captura realiza-se a partir do melhor ângulo disponível. O FOV de cada uma das câmaras, entre outros parâmetros intrínsecos ao dispositivo tais como coeficientes de distorção, e distâncias focais, devem ser considerados para obter um mapeamento entre os píxeis de profundidade e os de cor, obtendo assim uma nuvem de pontos colorida [OpenKinect, 2012].

Figura 2 - Cenário de interacção com o Kinect [Gadg, 2012].

2.2.2 Nintendo Wii

Nos anos 80 a Nintendo tentou, pela primeira vez, mudar o modo de interação dos seus produtos com a criação da Nintendo PowerPad. Contudo, apenas em 2006, com o lançamento da Wii, o público aderiu consideravelmente a este modo de jogo. Sendo em 2010 líder de vendas e ultrapassando a PSMove e o Microsoft Kinect. A Wii vem equipada com um sensor e um controlador semelhante a um comando de televisão, designado por Wii Remote, que utiliza um microcontrolador[3] ADXL330, que controla três acelerómetros em simultâneo, um para cada eixo, de forma a obter a velocidade e a posição da mão do jogador [Tanaka et al., 12].

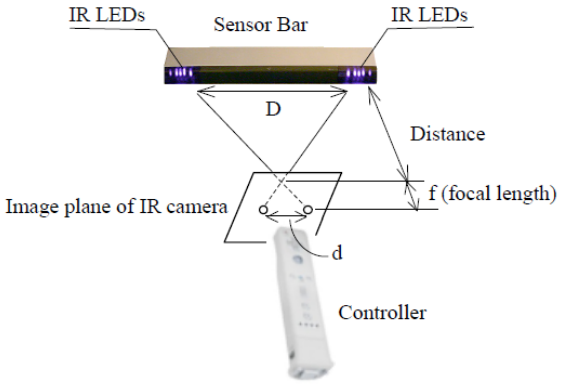


Figura 3 - Modo de detecção de movimentos do sistema Wii.

Em 2009 foi lançada uma nova versão da Wii, a Wii MotionPlus, trazendo melhorias relacionadas com a precisão da captura dos movimentos. Usando para o efeito um giroscópio, para melhorar a precisão dos dados obtidos das medições espaciais, e permitir a deteção de movimentos de rotação do controlador. Na Figura 3 encontra-se ilustrada a arquitetura do sistema *Wii* constituída por um sensor de projeção de luz IR e um controlador. O dispositivo *SensorBar* projeta dois feixes de luz IR, distanciados por uma distância D entre si. Por sua vez, o controlador, através de uma câmara IR, captura as luzes projetadas e obtidas no plano de imagem do controlador, determinando as coordenadas tridimensionais no espaço.

A Wii dispõe também de uma API.NET gratuita, designada por WiimoteLib [Wiimotelib, 2012], que oferece a possibilidade de desenvolver facilmente jogos para esta plataforma [Tanaka et al., 2012].

[3] Consiste num computador num *chip*, contendo um processador, memória e periféricos de entrada/saída.

2.2.3 PlayStation Move

A multinacional japonesa Sony lançou para o mercado, no ano de 2004, o dispositivo de captura visual baseado em gestos denominado Eye Toy. Este conseguiu alcançar alguma notoriedade, despertando o interesse do público por este novo modo de interação, apesar da baixa capacidade de captura de informação bidimensional de que dispunha.

A PlayStation Move inclui o Move Eye, que consiste numa câmara RGB (Figura 4) com uma resolução máxima de 640 por 480 píxeis, a funcionar a uma frequência de 60 a 120 fotogramas por segundo, com um conjunto de microfones integrados. Contém um controlador, designado por Motion Controller idêntico a um bastão com uma esfera luminosa na extremidade. O reconhecimento dos movimentos realiza-se com base na obtenção das coordenadas tridimensionais da esfera luminosa pela câmara RGB [Tanaka et al., 2012].

Figura 4 - Consola PlayStation 3 juntamente com o PlayStation Move (câmara e controlador) [Embedded Vision, 2012].

A perceção dos movimentos processa-se em função da perceção pela câmara RGB da esfera luminosa situada na extremidade do bastão. Para a obtenção das coordenadas tridimensionais da esfera é realizado um cálculo do conjunto de píxeis obtidos da imagem. Na (Figura 5) pode-se verificar, através da Equação 1, a variação de d relativa à projeção do diâmetro da esfera bastão D para um plano de imagem em função da sua distância. Assim, quando a esfera se aproxima da câmara, o valor de d aumenta; e quando se afasta, diminui [Tanaka et al., 2012].

$$Depth = D * f/d \quad \text{Equação 1}$$

De forma a melhorar a precisão na obtenção das coordenadas tridimensionais e corrigir situações que induzam em erros ou a ausência de informação do utilizador, provenientes da ocultação da esfera do bastão (por exemplo: quando o bastão está atrás das costas) ou erros originados pela inexistência da calibração do bastão, o Motion Controller, vem equipado com três sensores: um acelerómetro, um giroscópio e um sensor geomagnético. O acelerómetro e

o giroscópio funcionam segundo os três eixos cartesianos; já o sensor geomagnético serve apenas para auxiliar na correção de erros na orientação do bastão, calibrando-o segundo o campo magnético da Terra.

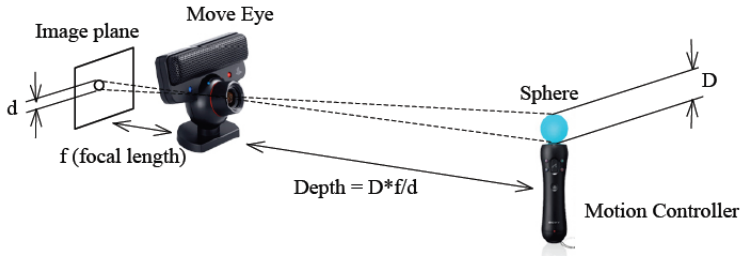


Figura 5 - Arquitectura de funcionamento do PlayStation 3 [Tanaka et al., 2012].

2.3 Dispositivos NUI

Além dos dispositivos NUI referidos na subsecção anterior aplicados essencialmente em consolas de videojogos, serão referidos outros não tão divulgados e conhecidos pelo público muito embora com grande importância pelo apoio prestado para o desenvolvimento de produtos NUI.

2.3.1 PrimeSensor™ Reference Design

A PrimeSense foi fundada em 2005 por Aviad Maizels, Alexander Shpunt, Tamir Berliner, Ophir Sharon e Dima Rais e é uma empresa Israelita líder em tecnologia NUI de conversão e perceção do mundo real para um ambiente em realidade virtual [Perlroth, 2010]. Esta empresa adquiriu grande relevo depois de participar no 'Projeto Natal' que deu origem ao dispositivo Microsoft Kinect, tendo a Microsoft anunciado publicamente a preferência pela sua tecnologia de leitura e tradução de movimentos para integrar no seu novo dispositivo, o Kinect [PrimeSense, 2012a].

Esta empresa, juntamente com outras duas companhias sem fins lucrativos, a Willow Garage e a Side-Kick, fizeram um esforço conjunto para a criação de uma plataforma padronizada para equipamentos com interface NUI, surgindo assim a plataforma OpenNI.

O *middleware*[4] NITE, lançado pela PrimeSense, localiza e deteta o corpo humano de cada um dos intervenientes presentes no cenário observado, separando a imagem de fundo dos

[4] É constituído por módulos e faz a mediação entre aplicações, movendo ou transportando dados entre programas de alto nível e proporcionando a sua integração com aplicações desenvolvidas em diversas linguagens de programação.

intervenientes e distinguindo cada um desses elementos. O NITE integra-se na camada de integração modular da plataforma OpenNI [Tscherrig, 2011].

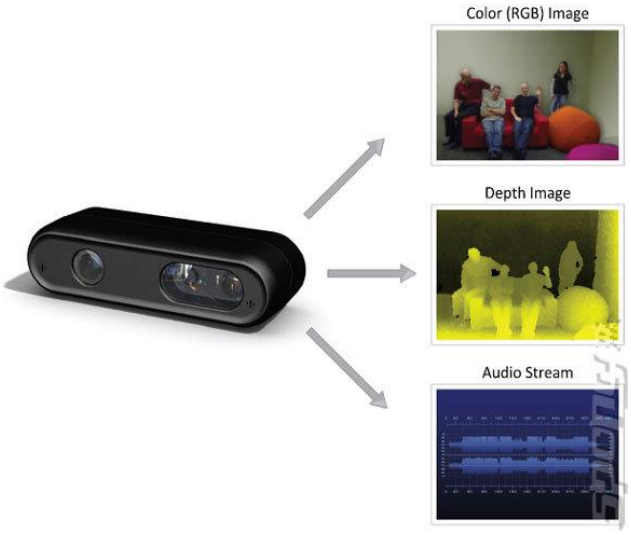


Figura 6 - Sensor NUI PrimeSensor [Spons, 2012].

O dispositivo comercializado pela PrimeSense, o PrimeSensor (Figura 6), recorre à projeção de luz estruturada, *Light Coding*, para codificar o cenário, projetando diferentes padrões ao longo deste, e captando os raios projetados, interpretando os padrões inerentes aos raios projetados e posteriormente refletidos nas superfícies dos objetos constituintes do cenário usando uma câmara Complementary Metal Oxide Semiconductor (CMOS). A luz captada ao longo do cenário é interpretada pelo microcontrolador PS1080 System-on-a-Chip (SoC[5]) (Figura 7), patenteado pela PrimeSense, que executa algoritmos complexos e interpreta os raios recebidos, representando o cenário observado através de coordenadas tridimensionais.

[5] SoC (em português: sistema-num-*chip*), assemelha-se a um microcontrolador e contém todos os componentes de um computador, ou outro sistema eletrónico, num circuito integrado.

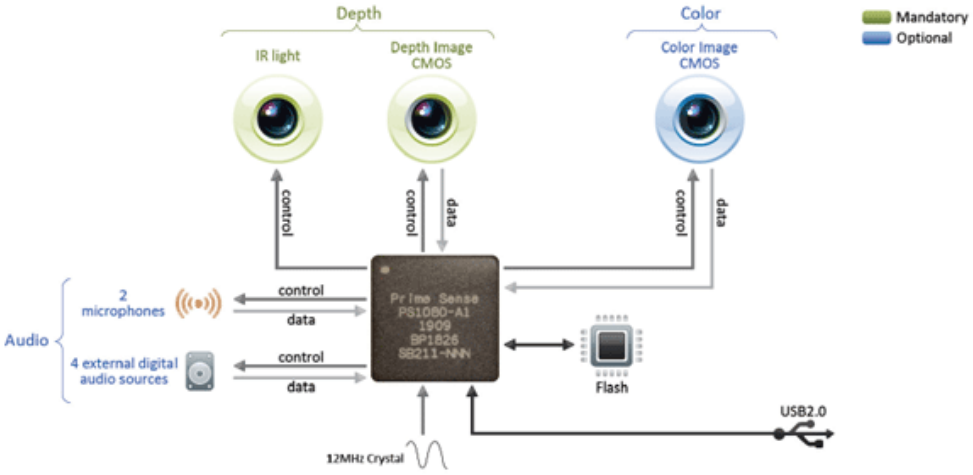


Figura 7 - Diagrama de blocos do PrimeSensor [Borglabs, 2012].

2.3.2 Leap

A Leap Motion tem prevista a comercialização do seu produto NUI entre Dezembro de 2012 e Janeiro de 2013. Este dispositivo (Figura 8), foi pensado para sistemas Mac OS X, Windows e Linux, dispondo publicamente de uma Software Development Kit (SDK) para permitir explorar o seu potencial. A tecnologia implementada no produto é ainda desconhecida. Contudo, a Leap Motion alega que o seu dispositivo é 200 vezes mais preciso do que o Microsoft Kinect, conseguindo alcançar o pormenor dos dedos das mãos e detetando movimentos de apenas 0.01 milímetros.

Figura 8 - Leap [Leap, 2012].

2.4 Conclusão

Neste capítulo foram descritos alguns diferentes dispositivos NUI, as suas principais especificações técnicas e aspetos inerentes ao seu funcionamento. Durante o processo de pesquisa e análise das bibliotecas existentes, foi possível constatar que, apesar da variedade de equipamentos existentes, o que apresenta maior versatilidade para o desenvolvimento de aplicações NUI parece ser o Microsoft Kinect. Este dispositivo dispõe de excelente

documentação, uma vasta comunidade de desenvolvimento e uma SDK de demonstração entre outras bibliotecas compatíveis disponíveis.

3 Captura de dados usando Microsoft Kinect

Ao longo deste capítulo, são abordados aspetos intrínsecos e extrínsecos ao dispositivo Microsoft Kinect escolhido para o desenvolvimento do protótipo. Serão relatados aspetos relacionados com o seu funcionamento, tais como a calibração das duas câmaras, o modelo matemático usado na medição da profundidade e a reconstituição das imagens de cor e de profundidade captadas. Consideram-se ainda aspetos essenciais para a correta reconstituição das imagens recebidas pelas duas câmaras e interpretação dos dados recebidos. Além disto, são também referidos fatores que induzem em possíveis erros e perdas na precisão na captura das imagens.

3.1 Calibração

A informação capturada pelas câmaras do Kinect ditam a precisão e a qualidade das imagens obtidas. Estas duas câmaras estenopeicas[6], dotadas de baixa resolução, funcionam com uma cadência de 30 fotogramas por segundo, resultando numa baixa precisão dos dados obtidos. A câmara RGB, juntamente com a de profundidade, obtém teoricamente uma nuvem de pontos com um total de 307200 pontos, que se refletem na realidade em apenas 265000 pontos, devido ao campo de visão segundo o qual os dados RGB são obtidos ser mais amplo em comparação com os dos dados da profundidade [Kjaer, 2011].

[6] Também conhecida por câmara *pinhole*, consiste numa maneira de ver uma imagem real, através de um pequeno orifício, sofrendo um movimento de inversão.

A qualidade da informação capturada está dependente da resolução destas duas câmaras e dos seus campos de visão. Estes encontram-se separados horizontalmente, sofrendo do problema designado como paralaxe, e que consiste em duas imagem semelhantes capturadas de diferentes pontos, embora próximos, terem diferentes perspetivas, o que implica realizar um conjunto de operações abaixo descrito na projeção geométrica, de forma a conciliar os dois tipos de imagens numa única. Atualmente estão disponíveis bibliotecas (por exemplo Point Cloud Library (PCL) [PCB, 2012]) que implicitamente mapeiam a informação de cor com a de profundidade, criando uma nuvem de pontos colorida.

Na calibração do dispositivo são considerados os seguintes parâmetros: a distância de focagem f, principais deslocamentos de pontos (x_0, y_0), o coeficiente da distorção das lentes $(\delta x, \delta y)$, o comprimento da base (distância entre a câmara IR e o emissor IR) b e a distância do dispositivo ao plano de referência Z_0.

A determinação correta destes parâmetros é essencial, de forma a evitar medições da disparidade incorretas e, consequentemente, originar uma fonte sistemática de erros nas capturas. Outro aspeto fundamental a considerar no processo de calibração passa pelo alinhamento dos sistemas de coordenadas das duas câmaras, descrevendo dois sistemas de coordenadas paralelos.

Os valores das variáveis em cima referidas são obtidos ao longo da fase de calibração. Contudo, b e Z_0 estão sujeitos às limitações inerentes à largura de banda do dispositivo para obter a profundidade de cada ponto k no espaço Z_k. O valor da disparidade é assim normalizado entre os valores de 0 a 2047, e substituído na Equação 9 que iremos ver mais adiante aquando do modelo matemático, aplicado na triangulação dos pontos, que através da derivação da profundidade permite achar a disparidade. A equação (Equação 2) é invertida (Equação 3) calculando os parâmetros Z_0, b, e f com base na disparidade obtida pelo modelo matemático e substituindo o valor de d da disparidade observada no espaço da imagem por $md' + n$ com d' normalizado [Khoshelham, 2011].

$$Z_k = \frac{Z_0}{1 + \frac{Z_0}{fb} d} \quad \text{Equação 2}$$

$$Z_k^{-1} = \left(\frac{m}{fb}\right) d' + (Z_o^{-1} + \frac{n}{fb}) \quad \text{Equação 3}$$

Para efeitos do cálculo da calibração são considerados os seguintes passos [Schwarz, 2011]:

1. <u>Projeção dos pontos bidimensionais dos pontos de profundidade para um espaço tridimensional.</u>

Consiste na passagem da imagem adquirida pela câmara IR (Figura 9), contendo a informação de profundidade da localização de cada pixel $x = (x, y)$, transpondo essa informação para um

sistema de coordenadas $X = (X, Y, Z)$ usando a Equação 4 do cálculo inverso para as coordenadas X , Y da projecção IR em que f_x, f_y, c_x e c_x são valores intrínsecos à câmara IR.

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} \frac{(x - c_x)d}{f_x} \\ \frac{(y - c_y)d}{f_y} \\ d \end{pmatrix} \quad \text{Equação 4}$$

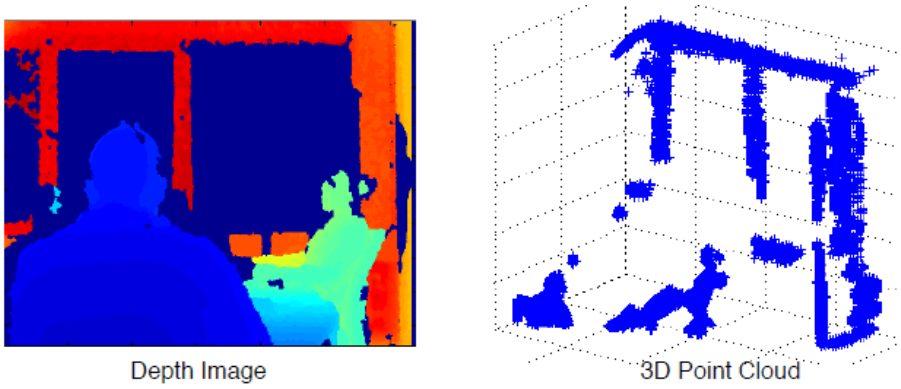


Figura 9 - Transposição das coordenadas bidimensionais de uma imagem em profundidade para um espaço de coordenadas tridimensionais.

2. Aplicação da transformação entre imagens

De acordo com Abdul Dakkak em [Dakkak and Husain, 2012], para calibrar os dois tipos de imagens, profundidade e RGB, é necessário realizar transformações rígidas entre elas. Os dados obtidos pelo Kinect, como referido, são valores que variam de 0 a 2047, representados por inteiros de 11 *bits*, dos quais 1 *bit* é reservado para sinalizar os píxeis nos quais a disparidade não é medida [Khoshelham, 2011]. Os restantes *bits,* que equivalem a um total de 1024, quantificam a informação de profundidade $Z_{i,j}$, convertendo-os em metros através da Equação 5 substituindo nos dados de profundidade $rd_{i,j}$ pelos valores da disparidade d assim como pelos respetivos índices i e j relativos às posições bidimensionais.

$$Z_{i,j} = \frac{1}{3.32788\, rd_{i,j}} \quad \text{Equação 5}$$

A Equação 6, representa a projeção da informação no espaço do mundo $W_{i,j}$. Encontram-se definidas as variáveis $c = \binom{339.308}{242.739}$ e $f = (0.00168289\ 0.00169193)$, que são constantes definidas internamente pela câmara de profundidade do aparelho e a informação de profundidade representada por $Z_{i,j}$. Estes parâmetros, aliados à Equação 6, realizam a projeção da informação de profundidade no espaço do mundo virtual:

$$W_{i,j} = f\left(\binom{i}{j} Z_{i,j} - c\right) \quad \text{Equação 6}$$

O passo que se segue consiste na transformação dos dados do mundo obtidos pela câmara de profundidade, nos dados do mundo compreendido pela câmara RGB $X_{(i,j)}$, multiplicando as suas matrizes de rotação e translação, como pode ser observado na Equação 7 A designa a transformação afim[7], que concilia a informação dos dois sensores.

$$X_{(i,j)} = A \begin{pmatrix} W_{i,j} \\ - \\ Z_{i,j} \\ 1 \end{pmatrix} \quad \text{Equação 7}$$

Por último, os pontos em profundidade são alinhados com as coordenadas RGB, usando os dois sistemas de coordenadas homogéneas definidos pela Equação 8, e utilizando como constantes $f = (529.215\ 525.563)$, $c = \binom{328.942}{267.4806}$ e $Z_{i,j}^{-1} = \frac{1}{X(i,j)\cdot z}$.

$$RGBZ_{i,j} = f\begin{pmatrix} \frac{X(i,j)\cdot x}{Z^{-1}} \\ \frac{X(i,j)\cdot y}{Z_{i,j}^{-1}} \end{pmatrix} + c \quad \text{Equação 8}$$

3. <u>Projeção dos pontos em profundidade numa imagem RGB.</u>

A projeção dos pontos consiste num processo de mapeamento dos píxeis da imagem RGB para a imagem em profundidade, transpondo para as coordenadas bidimensionais (x, y) desta as respetivas componentes de cores, colorindo a nuvem de pontos tridimensionais (Figura 10). Para isso, é essencial a correta orientação entre as duas câmaras, e uma calibração sistemática para assim corrigir sistematicamente erros no alinhamento entre as duas imagens [Khoshelham, 2011].

Segundo Nicolas Burrus, são necessários parâmetros inerentes ao *hardware* do equipamento. Valores como f, o centro da imagem e a distância focal de ambas as câmaras são essenciais para a correção da distorção existente entre as câmaras. Por último, procede-se ao mapeamento dos pontos de cor sob a nuvem captada pela câmara da profundidade, obtendo-

[7] Junção entre dois espaços vetoriais, segundo uma transformação linear Ax seguinda de uma translação $+b$. $x \rightarrow Ax + b$

se a nuvem de pontos colorida [Burrus N., 2011]. Este processo de rotação e translação será detalhado aquando da descrição do algoritmo de registo de nuvens de pontos Iterative Closest Point (ICP) (secção 3.6.1).

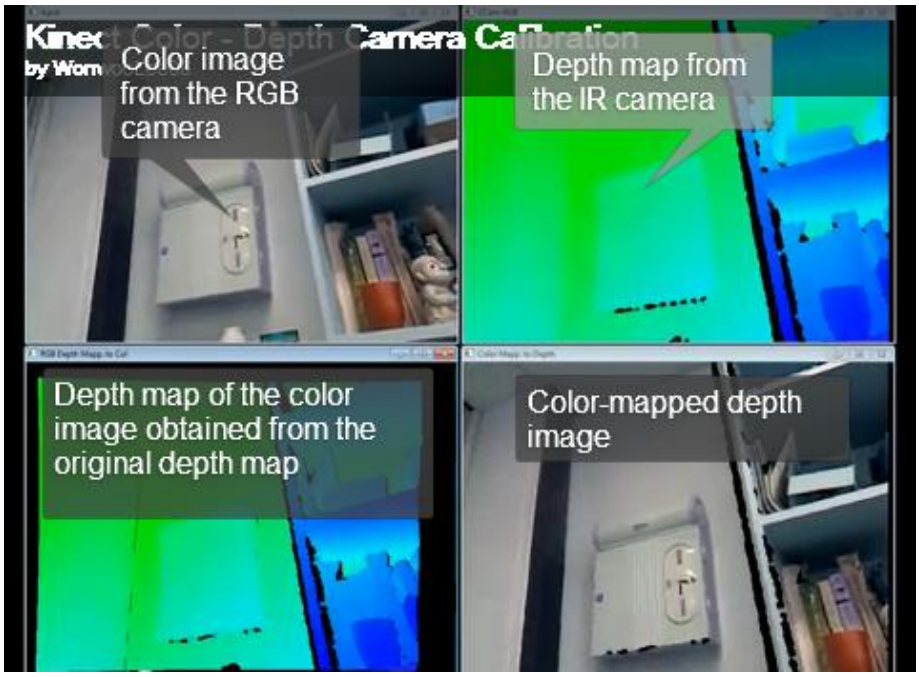


Figura 10 - Captura de vídeo elucidativa das junções possíveis entre as imagens RGB e as de profundidade. No canto inferior direito encontra-se a transposição da imagem de cor na de profundidade [WonwooLee06, 2012].

3.2 Modelo Matemático

A obtenção das posições das coordenadas dos pontos é realizada usando um método da projeção geométrica, aplicando um processo de triangulação para medição da profundidade dos pontos projetados. A Equação 2 anteriormente referida, está na base deste modelo para determinar parâmetros essenciais à calibração através da derivação da profundidade a partir da disparidade obtida. Na Figura 11 encontra-se representado o processo de medição da informação de profundidade, analisando a distância de cada um dos pontos. Os eixos usados são X, Y e Z, no qual Z encontra-se direcionado para a zona de projeção de pontos no cenário; X é-lhe perpendicular e paralelo a b, que representa a distância entre o projetor de luz IR e a câmara IR. O projetor e a câmara estão os dois direcionados para o ponto o no plano de referência, definindo um triângulo entre eles. Conforme a variação do ponto medido k, a distância ao plano de referência Z_k varia, bem como o valor da disparidade d. Assim, quanto maior o afastamento do ponto k ao plano de referência, maiores os valores da disparidade de D e Z_k [Khoshelham, 2011].

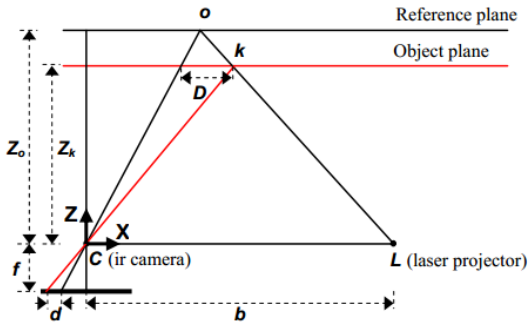


Figura 11 - Representação da relação da disparidade com a profundidade da medição do objeto [Khoshelham, 2011].

3.3 Medição da Profundidade por Triangulação

Em visão computacional, o uso de câmaras *pinhole* revela-se um aspeto essencial para melhoria da representação das imagens. O modelo de câmaras *pinhole*, também conhecido como modelo de câmara "ideal", melhora a qualidade da captura de imagem. Nestas câmaras, o centro da imagem é representado pelo que se encontra exatamente à sua frente, não existindo quaisquer lentes que causem distorção da imagem, fazendo assim com que as linhas traçadas pelos raios projetados sobre o cenário sejam verdadeiramente retas [Morvan, 2009]. A equipa Robot Operating System (ROS), decidiu realizar um estudo de modo a averiguar qual o nível de disparidade das câmaras do Kinect, podendo constatar que as suas características não divergem muito do modelo de câmara "ideal".

O processo de triangulação é baseado nas seguintes fases: emissão, reflexão e captura. Utilizando uma luz estruturada, esta é projetada pelo Kinect sobre o cenário filmado, sendo posteriormente refletida na superfície dos objetos que se encontram no cenário e, depois, capturada pela câmara IR, obtendo assim o padrão de luz codificada refletido. Isto deve-se ao sensor possuir, além de um padrão de luz codificado para codificação de cada posição do espaço, um padrão memorizado designado por plano de referência, referido anteriormente, definido a uma determinada distância conhecida pelo dispositivo. Este plano de referência serve como plano intermédio de comparação na variação dos pontos de luz mais próximos ou mais afastados desse, comparando a variação dos padrões recebidos e calculando a triangulação de cada ponto com base no modelo matemático.

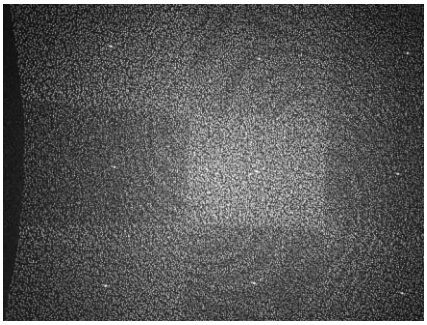


Figura 12 - Projeção de um padrão de pontos de luz ao longo de um cenário [kwc, 2012].

A projeção de pontos de luz consiste num feixe de luz que se divide em múltiplos feixes para codificar as várias posições do cenário. Ao longo deste são definidos diferentes planos de funcionamento, invisíveis ao olho humano, e que são determinados pela luz refletida ao longo do espaço, tendo como base o plano de referência (Figura 12). No caso dos pontos que ultrapassam o plano de referência, os raios refletidos que retornam encontram-se posicionados mais à esquerda. No caso do plano ser mais próximo, os raios irão retornar mais à direita, como mostra a Figura 13 [Kjaer, 2011].

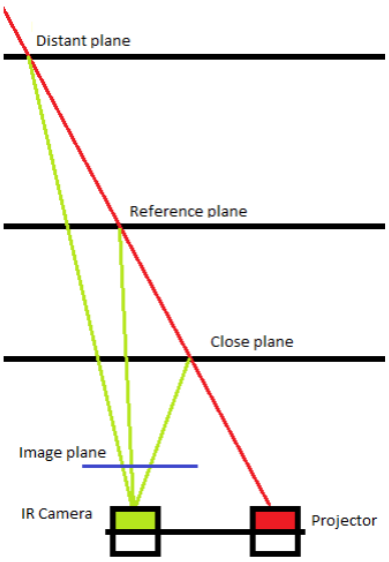


Figura 13 - Definição do plano distante e plano próximo através da luz estruturada projetada considerando como ponto intermédio o plano de referência [Kjaer, 2011].

Segundo testes realizados por Jacob Kjaer no seu artigo [Kjaer, 2011] com base na Equação 9 de Nicolas Burrus, é possível obter o valor da profundidade z em metros sabendo á priori o valor de disparidade d e as constantes intrínsecas do dispositivo -0.0030711016 e $+3.3309495161$.

$$z = 1.0/(d * -0.0030711016 + 3.3309495161)$$ Equação 9

O autor, nos seus testes, constata que, dos valores inteiros de 11 bits recebidos do Kinect, representados por d, apenas os valores que se situam entre 434 e 1084 são relativos a valores de medições, estando valores inferiores a 759 destinados a profundidades inferiores a 1metro, enquanto os restantes valores se destinam a profundidades superiores a 5 metros.

É possível verificar que a precisão do Kinect está bastante focalizada em distâncias entre os 60 centímetros e os 1.5 metros, enquanto distâncias superiores oferecem uma precisão menor e, consequentemente, uma representação mais débil dos pontos.

3.4 Fontes de erros e limitações

Algumas limitações do Kinect são intrínsecas e extrínsecas ao sensor IR. A primeira está relacionada com as imagens em profundidade obtidas do cenário e o aparecimento de "buracos negros" ao longo destas. Esta problemática prende-se com o facto de os cenários serem constituídos por vários obstáculos que se vão sobrepondo, como se pode ver na Figura 14. Apesar de a câmara IR conseguir visualizar a superfície mais afastada, não consegue captar a luz refletida nessa, pois a superfície mais próxima oculta parte da superfície mais afastada [Kjaer, 2011].

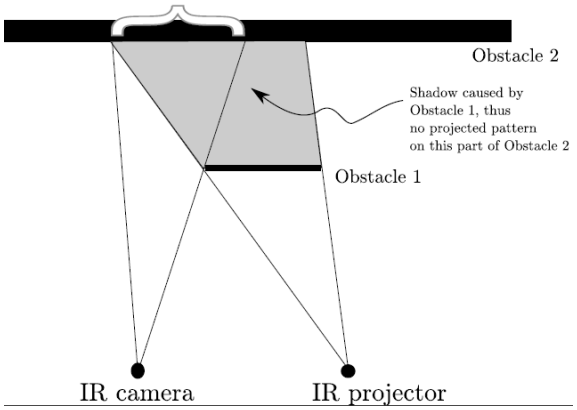


Figura 14 - Ilustração da sombra causada pela obstrução do Obstaculo1 [Media.zero, 2012].

Outra situação está intrinsecamente relacionada com a baixa resolução da câmara IR, originando uma gradual diminuição da fidelidade e da precisão dos dados captados com o aumento da distância. A Figura 15 representa a captura de uma bola de ginásio através de uma câmara IR situada na mesma posição em ambas as fotos, apresentando diferentes níveis de detalhe. A ilustração da esquerda encontra-se a mais de 2 metros de distância, enquanto a da direita situa-se a menos de 1 metro.

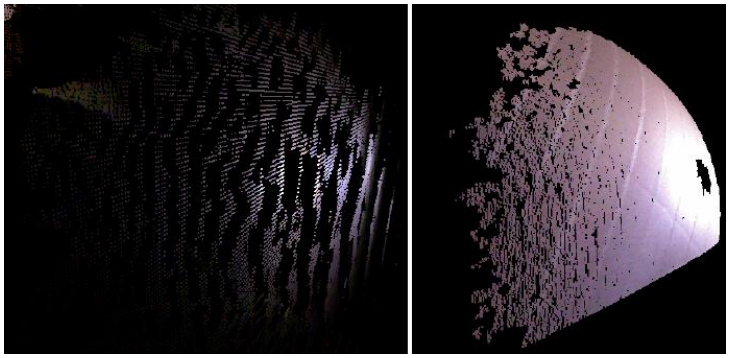


Figura 15 - Captura de uma bola de ginásio a distâncias diferentes [Kjaer, 2011].

Quando a superfície onde a luz é projetada é difusa, ou quando é atingida simultaneamente por outra fonte de luz externa (por exemplo: luz solar), dificultando o processo de descodificação, o sensor é induzido pelas reflexões incorretas, como se pode ver na Figura 16.

Figura 16 - Erro causado pela reflexão da luz solar numa bola de ginásio [Kjaer, 2011].

Por último, o facto de as duas câmaras estarem distanciadas, implica que, de cada vez que forem usadas, tenha obrigatoriamente de se executar o processo de calibração descrito anteriormente, de modo a que os dois tipos de imagens obtidos estejam corretamente alinhados.

3.5 Luz estruturada

Existem diferentes técnicas aplicáveis para a medição da profundidade de um cenário. São elas: através de micro-ondas, ondas de luz ou ultra-sons. Será abordada a técnica da luz estruturada, por ser o tipo de luz usado na elaboração deste projeto. As ondas de luz do tipo estruturada abrangem três diferentes técnicas de identificação dos pontos no espaço. São elas a multiplexagem no tempo, a codificação direta e a vizinhança espacial. Destas, a aplicada pelo Kinect foi a da vizinhança espacial, para codificar os pontos de luz emitidos pelo feixe de

luz IR. Os padrões de luz IR são gerados através de um código pseudo-aleatório para cada posição do cenário, projetando-os ao longo deste [Castaneda and Navab, 2011].

3.6 Algoritmo de Registo de Nuvens de Pontos

As nuvens de pontos são geralmente obtidas por sistemas de captura de digitalização de objetos ou superfícies, recolhendo um conjunto de pontos e representando-os através de um sistema de coordenadas cartesianas (x, y, z). Visto o estudo de nuvens de pontos não se enquadrar no âmbito deste projecto, será apenas descrito um algoritmo de registo de nuvens de pontos intitulado ICP. A recolha da nuvem de pontos pode ser realizada de duas formas distintas: manualmente ou através de bibliotecas já existentes publicamente disponíveis. As nuvens inicialmente recolhidas são sujeitas a perdas e ruído, pelo que devem ser tratadas através de funções de calibragem, filtragem e segmentação, de forma a obter uma nuvem de pontos final com precisão, como descreve Abdul Dakkak no seu artigo [Dakkak and Husain, 2012], no qual descreve a implementação manual desse conjunto de funções de tratamento.

3.6.1 Iterative Closest Point

O algoritmo ICP é aplicado no registo de nuvens de pontos através da digitalização tridimensional de objetos. Com a obtenção das coordenadas tridimensionais dos pontos, o algoritmo procede ao alinhamento das nuvens do modelo tridimensional usando métodos geométricos, malhas e a própria cor, como se verá posteriormente, de modo a obter o modelo tridimensional do objeto.

Segundo o estudo de Zhenggyou Zhang [Kjaer, 2011], através do mapeamento das coordenadas do objeto são obtidas duas nuvens de pontos iniciais de acordo com diferentes perspetivas (Figura 17). As duas nuvens de pontos recolhidas são sujeitas a sucessivas transformações, alinhando-as e criando correspondências aproximadas entre os pontos de ambas as nuvens.

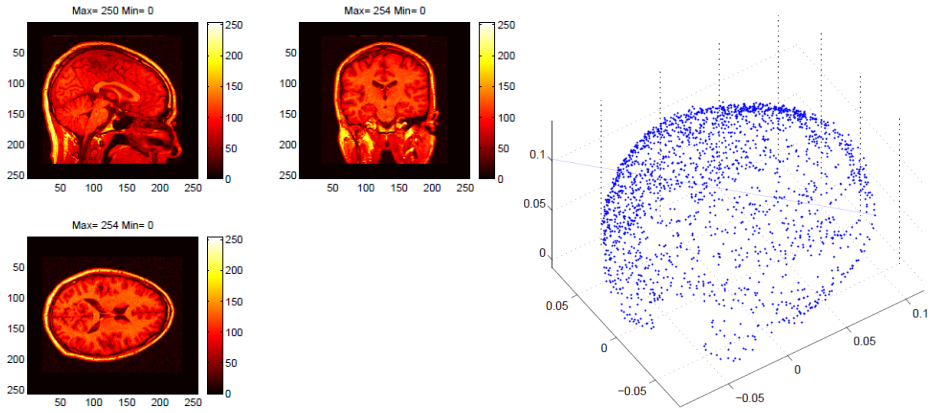


Figura 17 - Três perspetivas diferentes de um crânio humano (à esquerda). O mesmo crânio representado através de um modelo tridimensional resultante das três nuvens de pontos das diferentes perspetivas [Halchenko, 2008].

O método iterativo de convergência entre nuvens de pontos, implica duas nuvens de pontos, uma estática C_s e outra dinâmica C_m, fazendo com que estes dois conjuntos de pontos convergem num único conjunto. O conjunto C_m move-se para C_s, registando assim a nuvem C_m. Este processo de convergência implica que os dois conjuntos de pontos procurem, com base nas distâncias, pelos pontos mais próximos no outro conjunto, selecionando-os temporariamente e aproximando-os progressivamente de forma a juntá-los com a menor margem de erro possível. Depois disto, os pontos inicialmente temporários passam a definitivos e outro novo conjunto de pontos será selecionado para um futuro processo de convergência até que todos os pontos da nuvem fiquem registados. Esta metodologia descrita está resumida nos seguintes passos:

1. Através dos pontos p_m de C_m, encontrar os pontos correspondentes p_s em C_s, que correspondam aos vizinhos mais próximos;

2. Após p_m e p_s encontraremos os seus vizinhos, é calculada a transformação rígida através da rotação ortogonal $\boldsymbol{R}$ e a translação $\boldsymbol{t}$ (Equação 7) entre os pontos p_m e p_s, de forma a minimizar a margem de erro resultante da junção entre estes (Equação 10).

$$\min_{\boldsymbol{R},\boldsymbol{t}} \sum \|(\boldsymbol{R}p_{mi} + \boldsymbol{t}) - p_{si}\|^2 \quad \text{Equação 10}$$

3. Executar a transformação rígida sob C_m (Figura 18);

4. Repetir esta função até que todos os pontos de C_m, esteja totalmente registados.

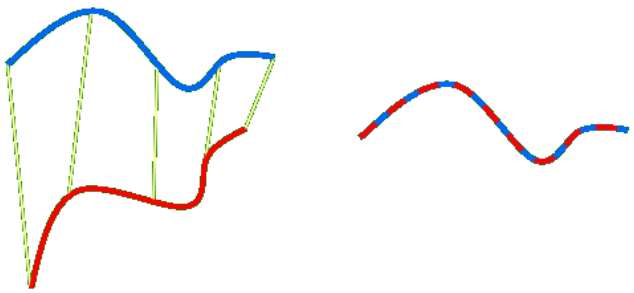


Figura 18 - A imagem à esquerda representa a correspondência entre pontos de duas nuvens de pontos diferentes. À direita encontra-se representada a transformação rígida dos pontos relacionados (à esquerda) representando a sua junção.

O ICP, dispõe ainda de várias extensões deste algoritmo, que assentam na versão base aqui descrita. Estas extensões permitem oferecer diferentes níveis de eficiência e são classificadas segundo as seguintes características [Rusinkiewicz and Levoy, 2001]:

1. A seleção dos pontos das duas nuvens;
2. O modo como a junção entre os pontos das duas nuvens se processa;
3. A influência de cada junção;
4. Com base na condição anterior (3), considerar se a junção deve ou não ser rejeitada;
5. Com base na junção entre os pontos das duas nuvens (2), considerar o modo de junção entre os pontos de forma a minimizar o erro de aproximação das duas nuvens.

3.7 Conclusão

Ao longo deste capítulo foi possível entender como as duas imagens são obtidas pelas duas câmaras e a forma como se relacionam. Foram apresentados os diferentes planos de referência, bem como os diferentes níveis de precisão inerentes a cada um deles. Com base nesta análise foi possível estabelecer: que o cenário aconselhado para operar corretamente com o dispositivo deverá conter um ambiente sem fontes de luz externas e usar o plano que garante maior precisão que se situa no plano de referência.

4 Software de desenvolvimento

No capítulo que se segue são introduzidas várias bibliotecas para desenvolvimento de aplicações NUI suas respetivas arquiteturas e as funcionalidades disponibilizadas. São também apresentados *middlewares* para comunicações de dados em redes informáticas, aplicados na transferência das coordenadas 3D do corpo do utilizador. Por fim, é referida a arquitetura da plataforma de desenvolvimento gráfico XNA aplicada no desenvolvimento do protótipo.

4.1 PrimeSense NITE

O NITE é um *middleware* que coloca à disposição um conjunto de Application Programming Interface (API) padronizadas, bem documentadas e acompanhadas de um número considerável de amostras de aplicações NUI. Esta componente permite realizar as seguintes operações [Villaroman et al., 2011]:

- Reconhecimento de gestos;
- Deteção do esqueleto e localização das articulações do corpo (Figura 19);
- Deteção da postura;
- Segmentação do utilizador e deteção de múltiplos utilizadores (Figura 20);
- Acesso à informação de profundidade e a dados de vídeo;
- Variação da calibração e suavização de funções de forma a melhorar o reconhecimento.

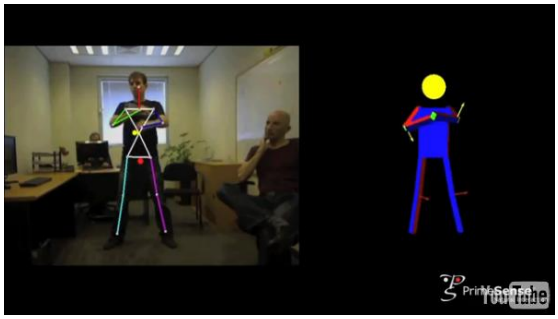


Figura 19 - Deteção das articulações e movimentos do utilizador [PStamirb, 2012].

O NITE dispõe ainda de algoritmos de visão computacional, bem como uma API NUI capaz de interpretar controlos baseados em gestos. Com isto, o utilizador pode ter uma experiência totalmente imersiva, permitindo-lhe interagir livremente com o computador ou outro dispositivo digital, usando apenas as mãos ou a totalidade do seu corpo [PrimeSense, 2012b].

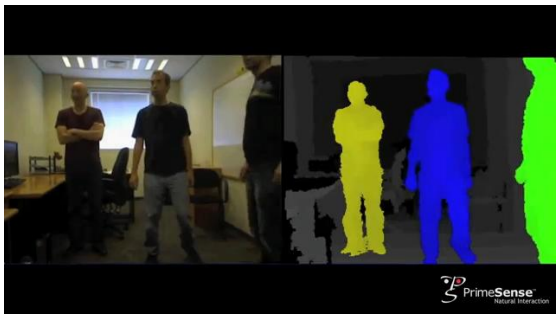


Figura 20 - Identificação dos vários intervenientes num cenário de simulação [PStamirb, 2012].

O NITE é integrado juntamente com a arquitetura OpenNI [PrimeSense, 2011c] e faz a separação dos elementos identificados na imagem de fundo, isto é identifica os vários utilizadores intervenientes num cenário em tempo real, analisando o cenário e interpretando os gestos e comandos de voz por estes produzidos.

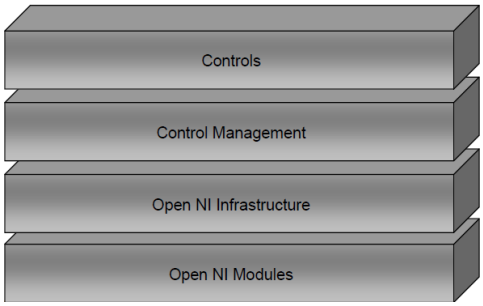


Figura 21 - Diagrama de blocos da infra-estrutura NITE [PrimeSense, 2011c].

Este *middleware* encontra-se subdividido nas camadas descritas na Figura 21. Começando na camada inferior, e seguindo para o topo, nesta camada inicial (OpenNI Modules) encontram-

se os módulos OpenNI *Gesture Generator*, *Hands Generator*, *Scene Analyzer* e o *User Generator* suportados pelo NITE. Na camada seguinte, *OpenNI Infrastructure,* encontra-se toda a estrutura modular da arquitetura OpenNI descrita mais adiante (Figura 22). Segue-se a camada *Control Management,* que recebe o fluxo de pontos encaminhando-os para a camada acima, *Controls* que, por sua vez, interpreta os dados recebidos e traduz os gestos captados em eventos [PrimeSense, 2011c].

4.2 OpenNI

A plataforma OpenNI surgiu em Dezembro de 2010, sendo fruto da associação de empresas sem fins lucrativos, como a Willow Garage, desenvolvedora de *hardware* e *software* livre para aplicações pessoais robóticas [Willow Garage, 2011]; a Side-Kick, líder no desenvolvimento de jogos e publicação especializada de vídeojogos baseados em NUI [SideKick, 2011]; e a fabricante de equipamentos informáticos ASUS, que comercializa equipamento NUI WaviXtion baseado na tecnologia PrimeSense [Rogers, 2011]. Estas empresas realizaram um esforço conjunto de forma a estabelecer uma plataforma padrão para equipamentos NUI, assegurando a compatibilidade, interoperabilidade e comunicação entre estes equipamentos, fazendo a captura e tradução da informação áudio e visual [OpenNI, 2011a].

A arquitetura, desenvolvida em C/C++ adequa-se a diferentes sistemas operativos, como o Windows (XP), Linux (Ubuntu superior ou igual 10.10), entre outras distribuições Linux, e Mac OS X[8]. O OpenNI tem como objetivo captar e traduzir a informação áudio e visual, integrando diferentes *middlewares* para serem usados pelas aplicações [Villaroman et al., 2011].

A sua infra-estrutura é segmentada por camadas, nas quais disponibiliza interface para módulos de dispositivos físicos e módulos para *middleware* de comunicação, associando-os à plataforma e acabando, assim, com problemas de incompatibilidades entre *middleware* e sensores, ou vice-versa. Com isto, passa a ser possível desenvolver aplicações e portá-las para funcionar com outro *middleware* ou dispositivo sem que sejam necessárias modificações suplementares.

As camadas que constituem a infra-estrutura OpenNI são (Figura 22) [OpenNI, 2011b]:

1. A camada aplicação funciona com base nos dados fornecidos da camada inferior e permite interagir com aplicações NUI alto nível;

[8] Sistema operativo proprietário baseado no kernel Unix , desenvolvido, fabricado e vendido pela empresa americana Apple.

2. Camada que representa as interfaces OpenNI, e os vários *middlewares* que podem ser associados à plataforma. Tem como objetivo processar os dados adquiridos pelos dispositivos e traduzi-los para serem interpretados pela camada superior;

3. Camada responsável por fazer a associação à plataforma dos diferentes dispositivos de captura de informação visual e áudio.

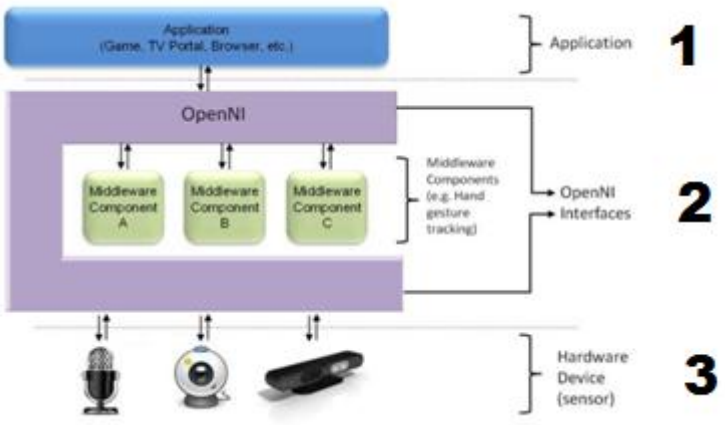


Figura 22 - Estrutura de camadas OpenNI [OpenNI, 2011b].

Os componentes suportados pelo OpenNI são [OpenNI, 2011b]:

Componentes de *hardware*:

- Sensor tridimensional;
- Câmara RGB;
- Câmara de infra-vermelhos;
- Dispositivo áudio (microfone ou conjunto de microfones).

Componentes *middleware:*

- Componente de análise total do corpo;
- Componente de análise ponto da mão;
- Componente de deteção de gestos;
- Componente analisador de cenas.

4.3 Microsoft Kinect SDK

O crescente interesse por parte da comunidade tecnológica e aficionada pela visão computacional, levou dois membros da Microsoft, Craig Mundie e Don Mattrick, a anunciarem para 21 de Fevereiro de 2011 a intenção em lançar uma versão não comercial de uma SDK do Kinect já na próxima Primavera. Porém, estava planeado para esta SDK o lançamento breve de um conjunto de ferramentas de desenvolvimento focado na pesquisa, para fins académicos e na comunidade entusiasta, disponibilizando-lhes acesso mais profundo a todas a capacidades do Microsoft Kinect, e só numa fase posterior o lançamento de uma versão comercial da SDK [Knies, 2011].

A 1 de Fevereiro de 2012, em Las Vegas, a Microsoft anunciou que o Kinect e respetivo *software* estariam inicialmente disponíveis apenas em doze países. Com a integração do Kinect com o sistema operativo Windows, a Microsoft pretende usar funcionalidades NUI como o suporte para gestos e voz, em sistemas embebidos para o controlo de sistemas inteligentes [Pinto, 2012a].

A 21 de Maio foi disponibilizada para *download* a versão 1.5 da SDK Kinect. Esta versão contêm um conjunto de ferramentas pensado para programadores e empresas e que lhes permite desenvolverem aplicações para este equipamento NUI. A SDK oficial Kinect, quando comparada com a sua versão *demo*[9]*,* vem com novas funcionalidades tais como, o Kinect Studio, que serve para a gravação e reprodução dos dados do Kinect, o uso de Human Interface Guidelines (HIG) para indicar as melhores práticas a seguir neste tipo de interfaces NUI, a localização do rosto, das expressões faciais e da rotação da cabeça. Além disto, inclui ainda amostras de código em C++, C# e Visual Basic e, por último, disponibiliza pacotes de reconhecimento da fala em diferentes linguagens. Esta versão comercializável veio permitir tanto a empresas como ao público em geral usufruírem de um conjunto de ferramentas para Windows direcionado para o Kinect [Pinto, 2012b].

4.4 OpenKinect - libfreenect

O OpenKinect, também publicado com a designação de libfreenect, foi desenvolvido na linguagem de programação C [OpenKinect, 2011] e é distribuído segundo uma licença livre. O seu surgimento partiu de um concurso promovido pela empresa Adafruit, como já referido anteriormente (secção 2.2.1). Este concurso visou fomentar o desenvolvimento de um *driver* para controlo do Kinect e que contemplava o vencedor com um prémio de $1,000. O participante de nacionalidade espanhola Hector Martin através de trabalho de engenharia reversa, conseguiu alcançar o objetivo e disponibilizar o seu *driver* para trabalhar com o Kinect,

[9] Versão de *software* inicial para demonstração.

em Linux. O trabalho iniciado por Hector Martin contou com a participação de novos elementos, que foram fazendo a manutenção, documentando e desenvolvendo os vários *wrappers* disponíveis deste projeto [OpenKinect, 2011]: Python; C Synchronous; Actionscript; C++; C#; Java JNI; Java JNA; Javascript; Common Lisp.

O *driver* acede através de uma ligação USB a funcionalidades do Kinect, tais como o acesso ao fluxo de dados das câmaras RGB e de profundidade, o controlo da rotação do motor e modificação do estado do LED para qualquer um dos seus 5 estados (Figura 23) [O'Reilly Answers, 2012].

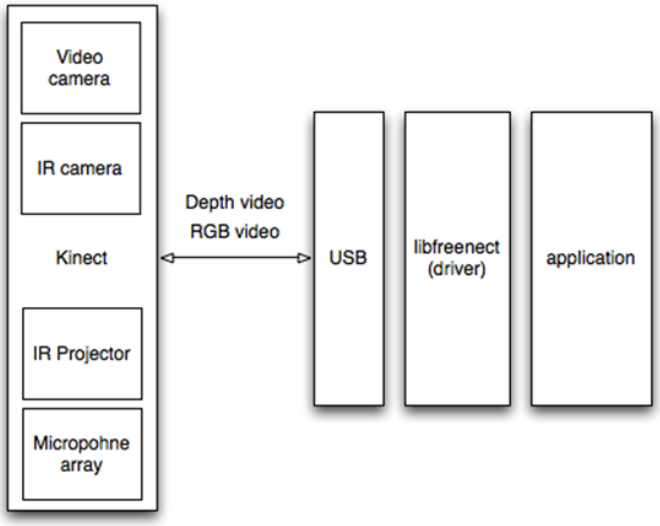


Figura 23 - Diagrama de alto nível do funcionamento do OpenKinect [O'Reilly Answers, 2012].

Esta biblioteca foi mais tarde modificada por Oliver Kreylos, que refere a sua modificação como, 'I didn't use any of his code, but the "magic incantations" that need to be sent to the Kinect to enable the câmaras and start streaming.' [Kreylos O. 2012]. Práticas estas que se revelaram essenciais, pois não possuía acesso à porta USB da consola Xbox360 para analisar e estudar o protocolo de comunicação.

4.5 Code Laboratories NI

O código de AlexP (secção 2.2.1) veio mais tarde a ser usado na primeira versão do Windows Kinect Driver/SDK - CL NUI Platform of Code Laboratories que começou oficialmente a ser distribuída e 8 de Dezembro de 2010 [Villaroman et al., 2011].

Esta biblioteca partiu do trabalho de investigação e desenvolvimento levado a cargo por AlexP e usado na sua obra, criando uma plataforma NUI bem documentada, integrando as componentes Áudio, Câmara e Motor (Figura 24).

O Code Laboratories (CL) NUI dispõe ainda de diversos *wrappers* nas seguintes linguagens: Actionscript; Java/Processing; C++; OpenCV e C#.

Os módulos que constituem esta biblioteca são os seguintes: Xbox NUI Áudio; NUI Motor; Acelerómetro.

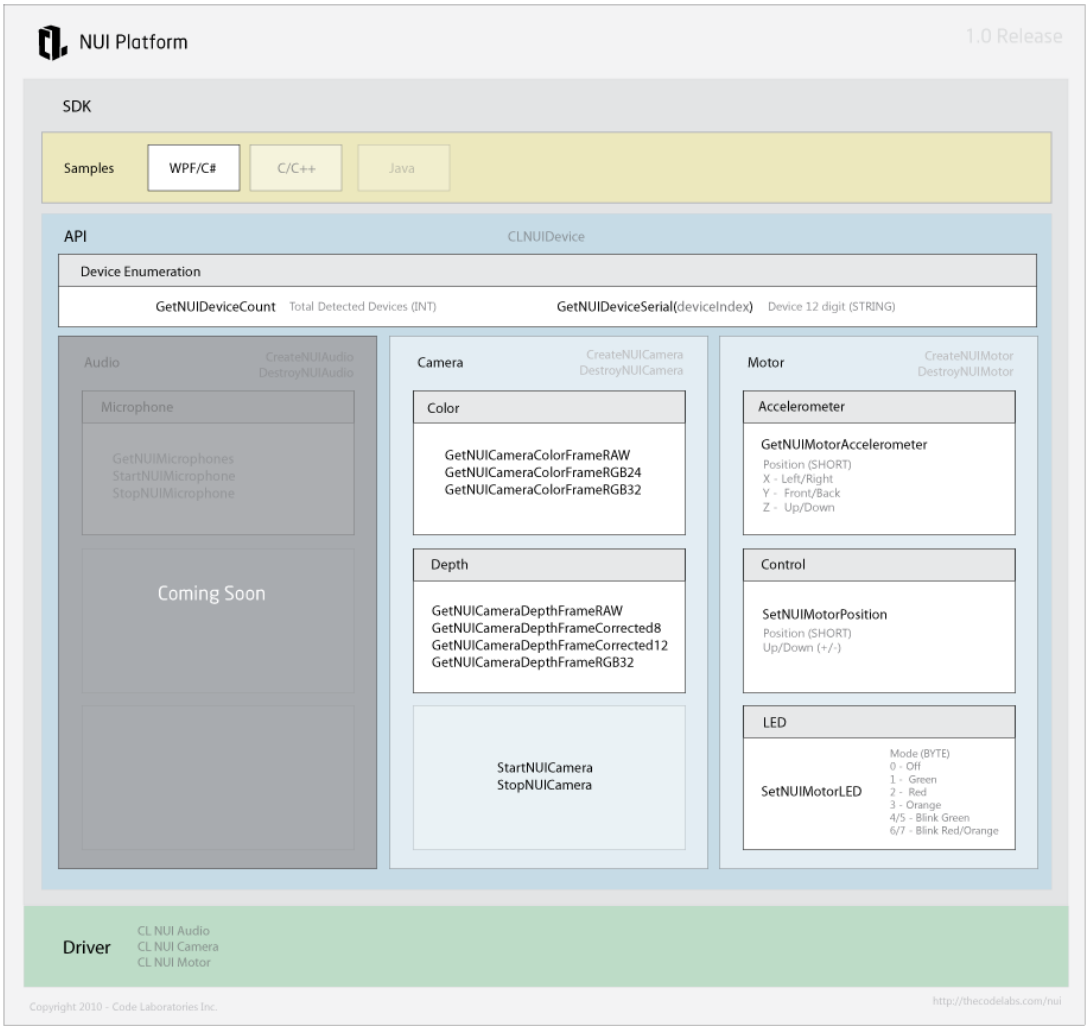


Figura 24 - Descrição geral da plataforma CL NUI [Code Laboratories, 2011].

4.6 Flexible Action Articulated Skeleton Toolkit

O Flexible Action and Articulated Skeleton Toolkit (FAAST) é um *middleware* de comunicação gratuito, destinado à pesquisa e a fins não comerciais. Este componente funciona juntamente com diferentes *drivers* NUI, para localizar e representar virtualmente as várias junções do corpo do utilizador usando um cliente Virtual-Reality Peripheral Network (VRPN) para enviar a informação através de uma rede de comunicação.

Figura 25 – Jogabilidade do World of Warcraft usando Microsoft Kinect com FAAST e OpenNI [Evansuma, 2012].

A biblioteca FAAST encontra-se atualmente na versão 1.0 e opera tanto com o sensor Kinect da Microsoft, como com o PrimeSensor da PrimeSense, permitindo ao utilizador executar comandos com base em gestos definidos que, ao serem captados, vão espoletar determinadas ações. Além disto permite, através de uma arquitetura de rede e de acordo com uma topologia cliente servidor, disponibilizada por VRPN, integrar o FAAST em motores de jogo já existentes, possibilitando a alteração do modo de interação com jogos existentes, em que até então era obrigatório utilizar o rato, ou o teclado, ou ambos para os jogar [FAAST, 2011].

4.7 Open Sound Control

O Open Sound Control (OSC) foi anunciado em 1997 por Wright e Freed como sendo um novo protocolo de comunicação entre computadores, sintetizadores de som e dispositivos multimédia concebidos para funcionar em redes de comunicações. [Adrian and Andy, 2009].

O seu crescimento foi fruto do desenvolvimento explosivo da Internet, começando na primeira especificação 1.0, que surgiu em 2002, e que se encontra atualmente na versão 1.1.

Este protocolo partiu de um grupo interdisciplinar designado por Center for New Music and Audio Technology (CNMAT) da University of California Berkeley Department of Music, grupo este que se foca na pesquisa e criação de meios de interação entre a música e a tecnologia, abrangendo diferentes áreas disciplinares em prol da criação de novas ferramentas para compositores e artistas [CNMET, 2011].

O protocolo apresenta uma estrutura protocolar bastante simples, usando um formato padrão específico, semelhante ao protocolo de comunicação entre instrumentos musicais Musical Instrument Digital Interface (MIDI). O MIDI comunica entre equipamentos musicais através de mensagens padronizadas, embora no caso do OSC, este também opera sobre redes de comunicações, fazendo com que a comunicação entre os equipamentos seja praticamente em tempo real, trocando mensagens de grande simplicidade protocolar, permitindo a

sincronização e o controlo e garantindo a interoperabilidade, flexibilidade e precisão na informação transmitida entre sistemas e dispositivos multimédia [OpenSoundControl, 2011].

Embora o protocolo se encontre na versão 1.1, desde a sua primeira especificação que era amplamente usado pela comunidade New Interfaces for Musical Expression (NIME) [NIME, 2011] para estruturas de controlo de programação e codificação de novos controlos gestuais. Este grupo dedica-se à criação de novas interfaces musicais, tendo desde cedo utilizado o protocolo OSC. O OSC serviu de base para o novo protocolo Tangible User Interface Objects (TUIO) [TUIO, 2011], usado para superfícies sensíveis a múltiplos toques e para o Gesture Description Interchange Format (GDIF) [GDIF, 2011], usado para troca de dados gestuais [Adrian and Andy, 2009].

4.8 Microsoft XNA Framework

A plataforma de desenvolvimento de vídeojogos lançada pela Microsoft veio substituir a *Managed DirectX* e convertê-la para um tipo de plataforma *.NET* [Microsoft MSDN, 2012a], designada XNA. O nome XNA não possui nenhum acrónimo, tendo o seu nome partido do projeto chamando *Xbox New Architecture*. Esta ferramenta foi oficialmente anunciada a 24 de Março de 2004 no *Game Developers Conference* nos USA, e a sua primeira versão, *Game Studio,* foi disponibilizada em 14 de Março de 2006 [Microsoft XNA, 2012].

O XNA foi a primeira API unificada para desenvolvimento de vídeojogos, e atualmente já dispõe de cinco versões *Game Studio*, facilmente acessíveis ao público entusiasta da área da computação gráfica, programadores individuais de vídeojogos ou académicos, permitindo-lhes desenvolver *software* para diferentes plataformas como *Windows*, *Xbox* e *Zune* gratuitamente. Outra grande vantagem que esta plataforma oferece aos desenvolvedores de *software* é a possibilidade de transferir as suas próprias criações não só para a Xbox360, mas também para o público interessado, e comercializá-la mediante a subscrição online no *XNA Creators Club*. Este site, além do que já foi referido, disponibiliza o apoio de uma vasta comunidade de desenvolvedores de videojogos para as diferentes plataformas [Microsoft MSDN, 2012b].

O *XNA Game Studio* integra um conjunto de ferramentas de apoio para facilitar o trabalho dos desenvolvedores de jogos de vídeo, incluindo componentes intuitivos para inserção de elementos gráficos e auditivos. A migração do *DirectX* para uma plataforma *.NET*, tornou o desenho da sua arquitetura homogeneizado relativamente às plataformas Microsoft existentes, como a *.NET Framework,* tornando possível integrar componentes entre elas.

Nesta plataforma encontra-se um conjunto de bibliotecas reutilizáveis, bem como um modo de compilação em código gerido, o qual é descrito em [Lobão et al., 2009] como um método que deixou inicialmente a comunidade entendedora no campo do desenvolvimento de videojogos bastante expectante e apreensiva quanto ao seu tempo de execução. Porém, este

método veio a revelar-se bastante eficiente, otimizando o tempo de execução de vídeojogos, executados sob uma máquina virtual a Common Language Runtime (CLR) [Rabello et al., 2008].

4.8.1 Arquitetura

A caixa de ferramentas XNA Game Studio inclui ferramentas que são incorporadas na respetiva versão compatível da interface de desenvolvimento Visual Studio, juntamente com um conjunto de ferramentas didáticas de aprendizagem.

A programação desta plataforma permite a execução do código em Managed Code[10], à semelhança do padrão seguido pelas plataformas .NET, o desenvolvimento modular de aplicações, e o uso de programação orientada a objetos, garantindo comodidade na integração de bibliotecas de outras plataformas .NET tais como a .NET Framework ou .NET Compact Framework (Figura 26) [Kvamme and Strøm, 2008].

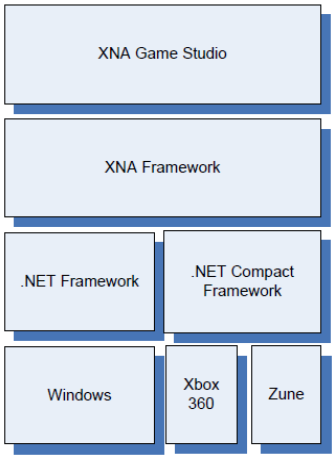


Figura 26 - Representação da estrutura de camadas da arquitetura XNA.

Na Figura 27 encontram-se detalhadas as camadas que compõem a Framework XNA. Na primeira camada encontramos as componentes relacionadas com o grafismo (DirectX), som, Cross plataform Audio Creation Tool (XACT) e entrada de comandos (XInput). Na camada seguinte, Core Framework, podemos identificar as várias funções essenciais à construção de um videojogo interativo: Graphics, Audio, Input, Math, e Storage. Na Extended Framework encontram-se os componentes Application Model e Content Pipeline, que representam os principais componentes para o desenvolvimento de uma aplicação XNA. E por fim, na última camada, Games encontram-se os vários conteúdos disponibilizados pela comunidade de desenvolvimento e pela Microsoft [Kvamme and Strøm, 2008].

[10] Termo usado pela Microsoft para designar código que é unicamente executado pela máquina virtual CLR.

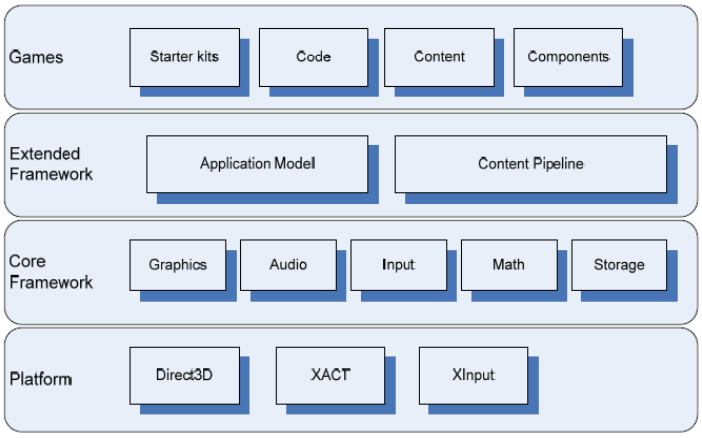


Figura 27 - Visão mais detalhada das camadas constituintes da plataforma XNA [Kvamme na StrØm, 2008].

4.8.2 XNA Framework

Cada *Game Studio* XNA assenta sobre uma plataforma *XNA Framework*, sendo esta biblioteca de classes responsável por disponibilizar o acesso às funcionalidades da *XNA Framework*. De um modo geral, os *Game Studio* estão organizados da seguinte forma [Microsoft MSDN, 2012d]:

- Microsoft.Xna.Framework.Audio: componente que faculta o carregamento e manipulação de áudio, trabalhado em projetos XACT e reprodução de ficheiros de som;
- Microsoft.Xna.Framework.Content: componente que acede ao *Content Pipeline* em tempo de execução, permitindo o uso de artefactos como imagens, modelos tridimensional e arquivos Extensible Markup Language (XML), previamente importados para o projeto;
- Microsoft.Xna.Framework.Design: aplicado na conversão entre tipos de valores;
- Microsoft.Xna.Framework.Graphics: componente que disponibiliza um conjunto de classes e funções que ajudam a tirar partido do *hardware* gráfico, bem como capacidades de aceleração 3D da aplicação;
- Microsoft.Xna.Framework.Input: conjunto de classes encarregue de receber e manipular a entrada de dados de periféricos como rato, teclado e *Gamepad*;
- Microsoft.Xna.Framework.Storage: componente que permite a leitura e escrita de ficheiros;

- Microsoft.Xna.Framework: componente essencial ao jogo como *Game*, *GameComponent*, *GameTime*, *GraphicsDeviceManager*, entre outros.

4.8.3 XNA Content Pipeline

Além dos componentes referidos, é possível inserir facilmente novos elementos (modelos tridimensional e texturas associadas) através da sua conversão para *Managed Code*, para que possam ser posteriormente utilizados e chamados pela aplicação em tempo de execução. Isto é possível aplicando ao ficheiro inicial um conjunto de transformações para que o artefacto seja convertido. São as seguintes:

- *Importer*: executa a transformação de um ficheiro no formato original para um formato intermediário, descrevendo-o num ficheiro em formato XML.

- *Processor*: depois do ficheiro gerado pelo *Importer*, o *Processor* executa a tradução do ficheiro temporário a partir do qual gera como resultados uma descrição do artefacto em código C#.

- *Compiler*: com o código C# do modelo gerado pelo *Processor*, o *Compiler* faz a sua compilação, gerando código no formato Microsoft Intermediate Language (MSIL), para permitir à aplicação usar o modelo.

- *Loader*: carrega o artefacto compilado quando solicitado pelo código em execução ou por comando espoletado pelo utilizador.

O *Content Pipeline* oferece grande flexibilidade e simplicidade na inserção de novos elementos numa aplicação XNA, permitindo ao desenvolvedor criar outros *Importers* e *Processors* além dos já definidos inicialmente pelo GameStudio, para diferentes tipos de ficheiros que de base não sejam suportados pelo XNA. É exemplo disto o motor de jogo *XNA Xen: Graphics API for XNA* [XEN, 2012], que possui dois novos tipos de *Processor* e *Importer* associados [Rabello et al., 2008].

4.8.4 Cross Plataform Audio Creation Tool

De modo a tornar a interação mais apelativa, é possível adicionar efeitos sonoros. Os dois modos disponíveis para reproduzir ficheiros de áudio derivados da API XNA Framework Audio são o *SoundEffect* e o Cross Plataform Audio Creation Tool (XACT).

O processo adotado baseou-se no uso do XACT. Este consiste numa interface de edição de conteúdo áudio que possibilita, também, a reprodução destes mesmos conteúdos através de eventos [Microsoft MSDN, 2012e].

4.8.5 Conclusão

Aquando do início deste projeto, a Microsoft apenas tinha disponível uma versão experimental da SDK do Kinect, que se destinava unicamente a testes e a estimular a exploração por parte da comunidade, apresentando ainda algumas limitações como o não necessitar de calibração, estabelecer deteção do utilizador com base num algoritmo que funciona com base em fotogramas, e problemas relacionados com paredes próximas, braços perto do corpo, oclusões, e só conseguir detetar o máximo de dois utilizadores em simultâneo, [Mehlmann et al., 2011]. Assim, para o presente trabalho, optou-se pelo uso da plataforma OpenNI. Esta opção deveu-se a fatores como o funcionamento em diferentes sistemas operativos, ser uma plataforma padronizada para qualquer equipamento, tratar-se de uma biblioteca de uso livre, sujeitando-a a constantes estudos, fazendo com que tenha grande maturidade e estabilidade.

5 Reconhecimento de caracteres

Neste capítulo será introduzida a importância da escrita à mão no quotidiano. Serão abordados diferentes tipos de reconhecedores e quais os se enquadram em cada tipo. Por último, será descrito o funcionamento e interpretação dos padrões de quatro algoritmos de reconhecimento.

A escrita à mão é uma técnica que tem vindo a ser desenvolvida ao longo dos tempos por várias civilizações em prol da sociedade humana, permitindo-lhes desenvolver a forma como comunicam entre si e registarem as suas ideias em suporte. Esta técnica consiste na associação gráfica de símbolos, de forma a expressarem ideias. Cada indivíduo possui uma forma própria de expor as suas ideias relacionando os símbolos.

A par do aparecimento deste tipo de aplicações surgiram também diversas bibliotecas que vieram melhorar o processo de desenvolvimento das mesmas. São exemplos a SATIN [Jason and James, 1999] e a Tablet PC SDK [Microsoft TablePC, 2012], ambas baseadas no desenho sob a tela (Figura 28).

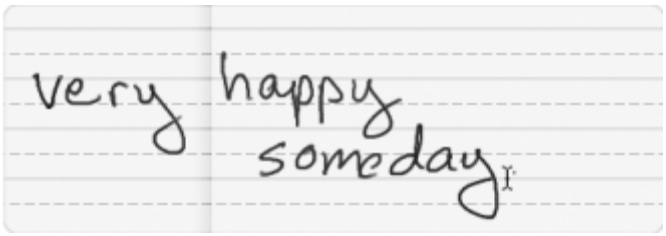


Figura 28 - Interface de desenho [O'Reilly, 2012].

Com o desenho à mão como forma de auxiliar na pedagogia infantil, têm emergido ferramentas de aperfeiçoamento como aplicações que se destinam ao desenvolvimento intelectual e motor (Figura 29). Estes estímulos visam melhorar progressivamente o controlo motor, aperfeiçoar os seus movimentos, a fluência com que desenham e a orientação pela

qual a elaboração dos traços se processa. Esta temática tem também vindo a ser estudada há vários anos por académicos para fins de reabilitação.

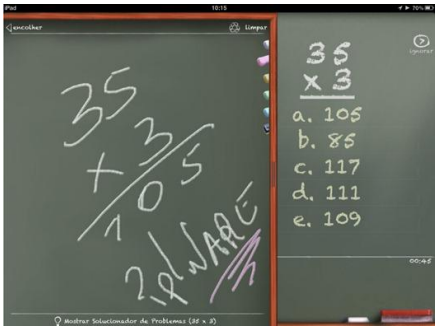


Figura 29 - Interface de escrita para aprendizagem MathBoard [Palasoftware, 2012].

No desenho de símbolos sobre uma superfície eletrónica de cristal líquido, a imagem manuscrita é digitalizada. A informação resultante é armazenada no dispositivo a cada movimento produzido. O processamento dos dados digitalizados realiza-se de acordo com dois diferentes modos de funcionamento. São eles o modo *off-line* e o *on-line*. No modo *on-line*, o armazenamento das coordenadas 2D é feito em função do tempo, considerando e armazenando informações relativas à ordem com que os traços são produzidos, empregando uma análise espacial do tempo. O modo *off-line* contempla apenas a análise da luminância espacial da imagem final do símbolo produzido [Plamondon et al., 2000].

5.1 Classificadores estatísticos

O reconhecimento de caracteres insere-se na ciência do reconhecimento de padrões, analisando características que compõem uma imagem, e decompondo-a de modo a extrair todas as particularidades que a caracterizam, comparativamente a outras amostras. Os dados das características recolhidas são discriminados em classes, e os seus valores representados através de vetores num espaço multidimensional.

A técnica de classificação estatística divide-se em dois diferentes grupos: a classificação supervisionada e a não supervisionada. Na supervisionada, o sistema usa padrões como fonte de conhecimento prévia, recorrendo a exemplos prévios para discriminação das classes. Assim, usando uma classificação supervisionada sobre o padrão a reconhecer, este é comparado com os restantes exemplos previamente conhecidos pelo sistema, classificando o padrão de acordo com o padrão do conjunto de amostras existentes que mais se assemelhe. A comparação de semelhanças com os padrões previamente existentes raramente é suficiente, de modo que a classificação deve também ter em conta outro fator, que é a proximidade entre os pontos.

O processo de classificação supervisionada está dividido em três etapas distintas que são:

1. Aprendizagem: este processo, referido anteriormente, baseia-se na aquisição de conhecimento por parte do sistema, usando-as como termo de comparação com futuras amostras;

2. Validação: Toussaint [Maxwell, 2012] descreve este processo com base em dois diferentes métodos de validação. São eles a auto-validação que, através da classificação do conjunto de aprendizagem do sistema, obtém a taxa de eficiência da classificação. Esta não permite, porém, determinar a capacidade de generalização de novos padrões. O outro método é a validação cruzada, considerado mais indicado por repartir os objetos por dois conjuntos, o conjunto de aprendizagem e o de validação. A classificação começa pelo conjunto de aprendizagem e passa ao de validação, encontrando neste último o valor da sua taxa de eficiência. Contudo, a validação cruzada é fortemente dependente do número de objetos conhecidos, e, caso este valor seja baixo, a validação fica dependente da seleção de objetos próprios dos dois conjuntos de aprendizagem e de validação, aos quais lhes extrai a taxa de eficiência com base na média da taxa destes. No caso do número de objetos ser elevado, a aprendizagem é então bem sucedida, e é definido um novo conjunto de aprendizagem que engloba todos os elementos pertencentes aos conjuntos de aprendizagem e de validação, usando-os para identificar novos padrões desconhecidos e determinando a capacidade de generalização de novos padrões;

3. Classificação: resulta do cálculo do grau de similaridade entre as amostras, tidas como conhecimento prévio e usadas na discriminação entre classes. A classificação é feita de acordo com dois tipos de classificadores: o tipo conexionista, que recorre a redes neuronais, como veremos mais adiante; e o tipo estatístico, que inclui dois tipos: o paramétrico e o não-paramétrico. Um classificador estatístico paramétrico (Figura 30) recorre à base de conhecimento estatístico apreendido fornecido pelas classes aprendidas; o não-paramétrico é mais simples e intuitivo, usando unicamente o fator distância até à classe mais próxima como fator de decisão/classificação. São exemplos de classificadores estatísticos não-paramétricos a distância euclidiana[11] e a distância de Mahalanobis[12].

[11] Distância entre dois pontos, obtida através do teorema de Pitágoras.
[12] A distância que estabelece uma correlação entre padrões distintos, permitindo estatisticamente determinar as semelhanças entre estas amostras.

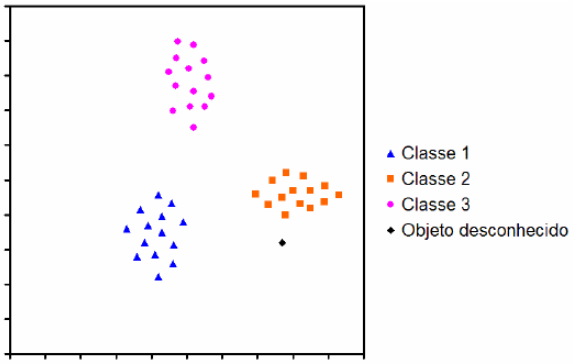


Figura 30 - Discriminação de três conjuntos de vetores em três classes distintas, com base na proximidade entre os pontos e recorrendo a uma classificação supervisionada [Mawell, 2012].

Um maior número de características para aprendizagem do sistema não implica necessariamente uma melhoria no desempenho do classificador. Pelo contrário, tornaria o sistema mais complexo e com maior custo computacional. Um grande número de amostras, além de prejudicar o desempenho do sistema, iria criar redundâncias que levam a que dois objetos aleatórios, ao serem descritos por um grande número de características, parecessem semelhantes, causados pela problemática da "Maldição da dimensionalidade" [13] e reduzindo desta forma a precisão do classificador. De modo a otimizar o processo classificativo, as características devem agrupar claramente objetos semelhantes e diferentes dentro de cada classe, facilitando a discriminação pelo classificador. Deve também ser encontrado um número de características que constitua um conjunto ótimo e que, assim, garanta um bom desempenho.

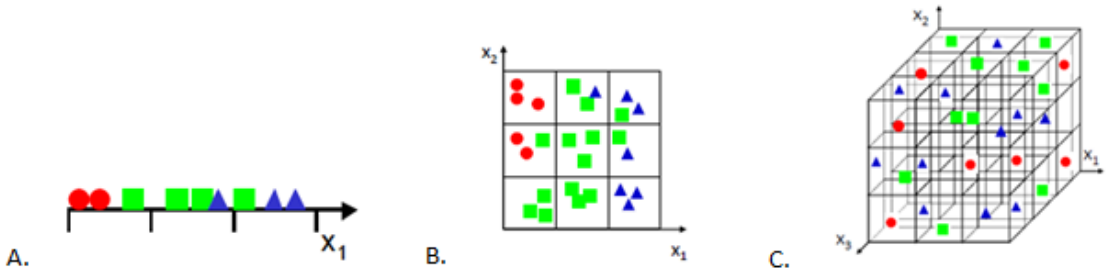


Figura 31 - A maldição da dimensionalidade; número de amostras aumenta exponencialmente com o aumento de dimensões.

A maldição da dimensionalidade está ilustrada no exemplo da Figura 31. Cada dimensão está dividida em 3 segmentos: na Figura 31 (a), uma dimensão, o número de amostras é 3^1; na (b)

[13] *Curse of dimensionality no* original em inglês.

duas dimensões, $3^2 = 9$; piorando na (c) três dimensões $3^3 = 27$, implicando um número de 81 amostras [Osuna, 2012].

Na classificação não-supervisionada não são usados quaisquer exemplos prévios, de forma que os vetores são representados de acordo com as suas semelhanças num determinado grupo. Com isto, a classificação é determinada segundo o grupo no qual o ponto relativo ao padrão se insere.

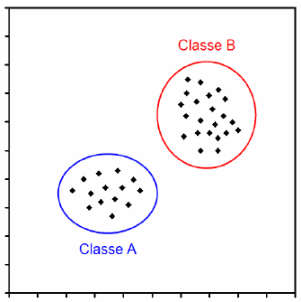


Figura 32 - Discriminação de dois conjuntos de pontos distintos, através de uma classificação não-supervisionada.

Na classificação de imagens, os dois tipos de processos classificativos descritos atuam sobre estas, de forma a quantificá-las, extraindo-lhes características que as definam e que possam ser medidas (Figura 32). Como se sabe, uma imagem é constituída por um conjunto de píxeis com diferentes cores e intensidades, definidos através de parâmetros possíveis de serem classificados. Assim, através da segmentação espectral de uma imagem bidimensional, esta é repartida e posteriormente sujeita a um processo de segmentação por limiarização, que simplesmente transforma a imagem definida por três componentes de cor R e G e B, em outra com apenas duas componentes. Isto é possível através da divisão de imagem em duas classes usando histogramas[14], para determinar qual das classes é preponderante com base no valor das duas componentes [Maxwell, 2012].

[14] Consiste numa representação gráfica de distribuição de frequências dos pesos de variação de um conjunto de dados.

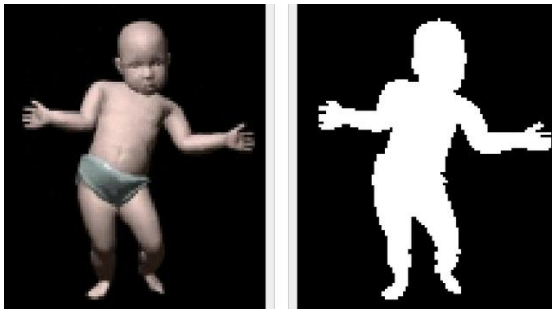


Figura 33 - Exemplo de imagem limiarizada [Skeletonmaker, 2012].

No reconhecimento de padrões, a extração e seleção de informação, assim como a sua representação, são aspetos cruciais a considerar. Por isso, antes de abordar os algoritmos utilizados no projeto para o processo de reconhecimento de padrões, serão abordadas algumas das variantes que estiveram na base do aperfeiçoamento e evolução desses métodos.

O método estatístico supervisionado Linear Discriminant Analysis (LDA) consiste numa função estatística usada na aprendizagem da máquina, que recorre a uma classificação linear para identificar as melhores características, que descrevem os dados após a sua classificação, reduzindo assim o problema da dimensionalidade e melhorando o reconhecimento das classes. Os dois métodos LDA e o Principal Component Analysis (PCA) de Karl Pearson criado em 1901 [Souza, 2012a], assemelham-se pois ambos procuram as combinações lineares que melhor descrevam os dados e de forma a corrigir o problema da dimensionalidade. O método estatístico não-supervisionado PCA usa um processo matemático para encontrar um pequeno conjunto de variações de um conjunto maior de componentes de uma ou mais classes, considerando unicamente a maior variedade possível que os dados apresentem, de forma a definir com eles um grupo de componentes principais, ordenando-os por ordem decrescente de acordo com a sua variedade. Contrariamente ao PCA, que é um método não-supervisionado, logo não usa informação prévia, o algoritmo supervisionado LDA não baseia a sua análise unicamente nas diferenças dos componentes, mas também nas semelhanças existentes entre as classes, discriminando-as com base nesse conhecimento prévio e na procura pela transformação linear, de forma a maximizar a variância entre as classes. Quer isto dizer que o LDA procura linearmente características que melhor descrevam as diferenças entre os dados, à semelhança do PCA, embora neste com conhecimento previamente adquirido [Souza, 2012b].

Ambos os métodos são algoritmos lineares que operam sob distribuições de dados lineares. Porém, com o aparecimento da técnica Kernel, passou a ser possível estabelecer um elo de ligação entre os algoritmos lineares e os não-lineares. Com esta ferramenta matemática é possível mapear os dados num espaço dimensional para outro maior, assim como a capacidade de estruturar melhor esses mesmos dados.

O uso desta técnica $\mathrm{k}(\mathrm{x}, \mathrm{y})$ faz com que seja desnecessário o mapeamento das características existentes no espaço de entrada para um espaço de maior dimensionalidade φ, bem como realizar o cálculo do produto interno desses dados, visto que, indiretamente, esta técnica

efetua-o no próprio espaço de entrada sobre as posições do vetor $(\varphi(x), \varphi(y))$. A Equação 11 mostra a valia desta técnica, que recorrendo à substituição de todos os produtos escalares por uma função Kernel obtém o produto interno no espaço de características não – linear [Souza, 2010a].

$$k(x, y) = (\varphi(x), \varphi(y)) \quad \text{Equação 11}$$

A técnica Kernel dispõe de um conjunto de funções de configuração de forma a adequar e definir o hiperplano ótimo da função de decisão não-linear de acordo com o que se pretende modelar. Apesar da variedade considerável de funções aplicáveis existentes serão apenas abordadas as três seguintes, pelo facto de terem sido contempladas no projeto de dissertação:

- Kernel Linear

Usa uma equação mais simples, que resulta da soma do valor opcional da variável c que controla a margem de erros no processo de aprendizagem, juntamente com o produto interno entre x e y e T usado como valor da tolerância de convergência. Esta equação modifica o modo de operação do algoritmo Kernel de um modo não-linear para linear.

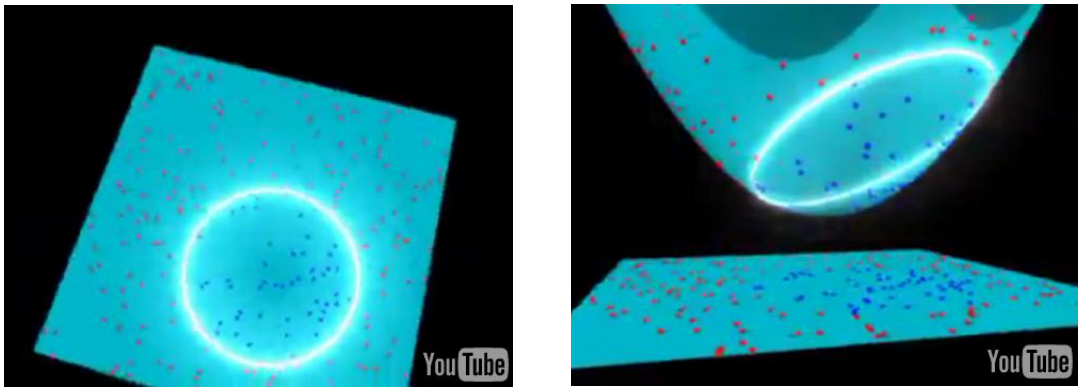
- Kernel Polinomial

Esta equação adequa-se a situações nas quais os dados se encontram normalizados.

- Kernel Gaussiano

A equação gaussiana é umas das equações usadas mais frequentemente pela sua permissividade e flexibilidade com base de uma função radial[15], generalizando-se para variados tipos de aplicações.

[15] Função que depende exclusivamente da distância de um ponto ao centro.



(a) (b)

Figura 34 - Transposição dos dados mapeados no espaço de entrada para um espaço de maior dimensão usando uma função Kernel [udiprod, 2012].

Na Figura 34 encontram-se as características representadas por intermédio de vetores de pontos representados no espaço de entrada. Na imagem (b), os dados representados por pontos azuis e vermelhos encontram-se mapeados não-linearmente usando a função Kernel Polinomial num espaço de características de maior dimensão.

De modo a complementar o algoritmo linear PCA e torná-lo mais eficiente, foi aplicada a técnica Kernel à sua versão linear base derivando o Kernel Principal Component Analysis (KPCA) que, à semelhança do PCA, extrai as características, embora no KPCA o conjunto não-linear de componentes principais passe a ser mapeado num espaço multidimensional de maior dimensão [Souza, 2012c].

5.1.1 Kernel Discriminant Analysis

À semelhança do PCA, o algoritmo LDA possui uma extensão não-linear, denominada KDA, que, à semelhança do LDA, procura maximizar a variância entre as classes e minimizá-la dentro das classes. A Figura 35, apresenta duas classes que representam dois conjuntos de características de entrada, difíceis de discriminar diretamente usando a função não-linear KDA por se encontrarem no espaço de entrada. Recorrendo à técnica Kernel esta vai implicitamente mapear os dados para um espaço de características com maior dimensionalidade e, de seguida, por se tratar de um espaço de alta dimensionalidade onde algoritmos lineares não são praticáveis, a técnica Kernel fica encarregada de calcular o produto interno e obter a distribuição linear no espaço de entrada (lado direito).

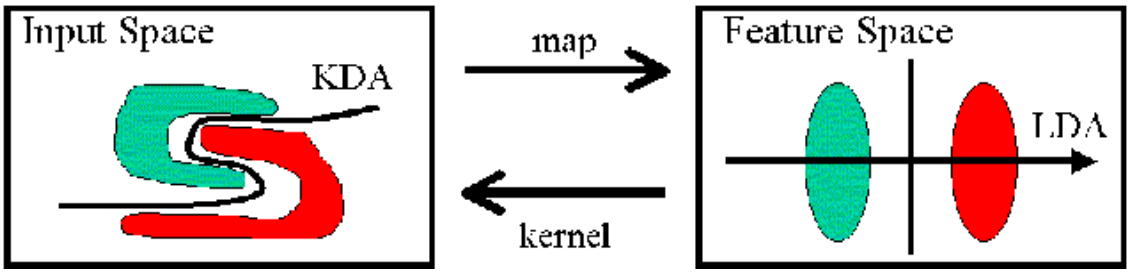


Figura 35 - Ilustração elucidativa do funcionamento da técnica Kernel aplicada através do algoritmo estatístico supervisionado KDA a um conjunto de características (verde e vermelho) no espaço de entrada [Yongmin et al., 2001].

A Equação 12, representa a procura realizada pelo KDA para encontrar a transformação, maximizando a variância entre as classes e minimiza-a dentro destas. Nesta equação, K_c representa a matriz da classe c, I representa a matriz identidade, I_c o número de amostras relativas à classe c e 1_{lc} a multiplicação de todas as matrizes de entrada $l_c * l_c$ [Souza, 2012f].

$$S_W = \sum_{c=1}^{C} K_c(I - 1_{lc})K_c^T \quad \text{Equação 12}$$

5.1.2 Support Vector Machines

Nos anos sessenta, o uso de técnicas Kernel era amplamente divulgado e aplicado, fazendo com que houvesse um aumento pela procura deste tipo de técnicas de análise e reconhecimento de padrões, levando à notoriedade do método de aprendizagem supervisionada SVM [Souza, 2012d].

O método linear descoberto por Vladimir Vapnik em 1963, opera num espaço Kernel Hilbert[16] de forma a proceder às classificações e regressões diretamente no espaço de entrada dos pontos. Esses pontos são facultados previamente com um conjunto de características para aprendizagem, definidos através de vetores z e apresentados como pontos no espaço com os seus respetivos pesos w. Destes, apenas dois são selecionados de forma a estabelecer um hiperplano com a maior distância entre os dois pontos, definindo duas classes de decisão (Figura 36). A classificação binária dos pontos descrita pela Equação 13 irá depender do posicionamento destes ao longo do hiperplano estabelecido. A variável b representa o produto escalar, z são os vectores de suporte aplicados na equação e respetivos valores w de 0 até N [Berwick and Idiot, 2003].

[16] Espaços definidos por funções Kernel.

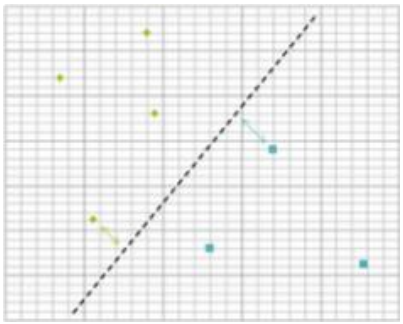


Figura 36 - Classificação de um conjunto de vetores de pontos em duas cores (amarelo) e (azul), usando como limite de decisão entre estes a definição de um hiperplano, com base a maior distancia dos pontos de aprendizagem mais próximos [Souza, 2010e].

$$F(x) = \sum_{i=1}^{N} \boldsymbol{w}_i(\boldsymbol{z}_i, x) + b \quad \text{Equação 13}$$

O processo de regressão SVM consiste na definição de dois limites de perdas ε, um superior e um inferior, com origem no hiperplano (Figura 37), para a classificação de um conjunto de dados. Os dois limites devem-se ao facto da existência de possíveis perturbações no momento de aquisição dos dados, de forma que os valores representados devem estar espaçados por uma margem máxima, para assim melhorar a generalização das amostras, ignorando possíveis erros e oferecendo vantagens a nível de desempenho computacional. Com isto, todos os valores que se encontrem dentro da margem máxima de erro definida pela margem superior e inferior são classificados com o valor zero [Support Vector Machine Regression, 2012].

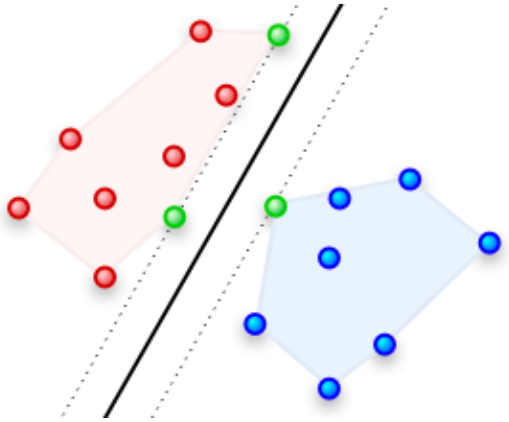


Figura 37 - Ilustração dos limites ε de um SVM regressivo [Geometry of Suport Vector Machines, 2012].

Aplicando a técnica Kernel ao algoritmo supervisionado SVM, os dados de aprendizagem são transportados de um espaço de entrada com as características originais para um espaço de maior dimensão. Porém, os dados não são lá mapeados. Assim, à semelhança da sua versão linear, este algoritmo está intrinsecamente dependente da obtenção dos valores do produto

escalar dos vetores, de modo que através da substituição dos produtos escalares obtém-se a função não-linear Kernel, (Equação 11) [Souza, 2010e]. Usando a classificação com base na função Kernel, esta faz uma classificação com base no hiperplano definido no espaço de maior dimensionalidade dos dados lá mapeados, e calcula o produto escalar dos vetores no espaço de maior dimensionalidade, transpondo o seu resultado para o espaço de entrada.

Como já foi referido, o SVM faz uma classificação binária com base num hiperplano, estabelecendo com isso apenas duas classes de resultados. De forma a corrigir esta limitação, e permitir uma maior generalização do número de classes, uma das possíveis abordagens a este problema passa pela classificação das múltiplas classes. Esta abordagem consiste na permuta das várias classes a classificar, eliminando redundâncias entre elas (exemplo classe: A* A), bem como combinações duplicadas (exemplo classe: A*B=B*A) [Souza, 2010e].

Após a obtenção do número de pares de combinações a decompor, são criadas tantas máquinas SVM quanto o número de pares existentes. O processo de decomposição de um problema que envolva múltiplas classes pode ser abordado seguindo as seguintes metodologias:

Decomposição um contra um:

Realiza-se com base num número de votações que cada classe obtém, de forma que a que obtiver um número maior de votações é a classe vencedora.

Decomposição um contra todos:

Este método estabelece uma máquina SVM para a classe, e com base no valor do classificador de cada uma das máquinas, associa automaticamente o valor máximo obtido à classe de um novo padrão, o que leva crer que seja pouco confiável [Gonçalves, 2009].

5.1.3 Sequential Minimal Optimization

O Sequential Minimal Optimization (SMO) é um algoritmo usado para aprendizagem de máquinas de vetores de suporte para resolver grandes problemas de otimização de programação quadrática[17]. Este algoritmo foi inventado por John C. Platt em 1998. O SMO consiste num processo iterativo através de um ciclo interno, partindo grandes problemas de programação quadrática em pequenos sub-problemas, para depois serem analiticamente resolvidos, evitando operações de programação quadrática que consumam demasiado tempo [John, 1998].

[17] Um problema de otimização linear limitado com uma função quadrática definido por uma parábola.

5.2 Reconhecedor $1

O $1, contrariamente aos algoritmos abordados anteriormente, destaca-se por ser um algoritmo conciso, simples, fácil de ser combinado com diferentes linguagens e interfaces, como mostra a aplicação deste algoritmo em Javascript numa página de universidade de Washington [Wobbrock et al., 2007a].

O $1 é um método estatístico não-supervisionado com excelente taxa de reconhecimento. A sua qualidade foi demonstrada no Best Windows Game, em 2008, integrando a componente de vídeo do jogo vencedor [Anthony and Wobbrock, 2007a].

O reconhecedor funciona com base em geometria simples e trigonometria. E a sua simplicidade e funcionalidade faz com que possa ser implementado em sistemas de prototipagem rápida. Neste são definidos dois conjuntos, o conjunto de pontos desenhados no momento para reconhecimento e designado de conjunto candidato C, e as várias amostras de pontos existentes T_i que servem como modelos de comparação. O $1 caracteriza-se pela sua total invariância no que respeita à orientação dos gestos, fazendo, por exemplo, que os símbolos '>' e '<' sejam considerados como sendo o mesmo símbolo no processo de reconhecimento, apesar das suas diferentes orientações.

A comparação entre os dois conjuntos C, e T_i usa o calculo da distância euclidiana (Equação 14), para determinar a melhor aproximação dos conjuntos de pontos existentes em T_i e que oferecem melhor alinhamento com o conjunto candidato C desenhado percorrendo de $k = 1$ até N. Assim, com base nesta aproximação, é obtido o valor da menor distância d_i entre os vários conjuntos testados e quantificados usando um sistema de pontuações, sendo que o conjunto com maior pontuação é o conjunto selecionado [Wobbrock et al., 2007b].

$$\mathrm{d_i} = \frac{\sum_{k=1}^{N} \sqrt{(C[k]_x - T_i[k]_x)^2 + (C[k]_y - T_i[k]_y)^2}}{N} \quad \text{Equação 14}$$

Antes do processo de procura e alinhamento do conjunto C, o caminho de pontos definidos por este é sujeito a quatro operações. A primeira passa por tornar os pontos de C equidistantes, reamostrando os pontos segundo um processo iterativo, no qual o distanciamento dos pontos varia consoante esse incremento. No segundo passo, procura rodar C através do ponto definido no centróide[18] do caminho dos pontos e pelo ponto inicial, seguindo-se a sua rotação para o ângulo indicativo (0°). Além de se encontrar o ângulo indicativo do conjunto C, são também procurados os ângulos indicativos dos conjuntos T_i

[18] Ponto no interior de uma forma geométrica que define o seu centro geométrico.

aquando do seu carregamento, permitindo assim aperfeiçoar a procura do conjunto que melhor se alinha usando o método de alinhamento Golden Section Search (GSS)[19].

O terceiro passo consiste em redimensionar não uniformemente o caminho de pontos para um quadrado tido como referência com um comprimento dos lados igual em todas as amostras de T_i. Por fim, realiza-se a translação do centróide de C para a origem $(x, y) = (0,0)$. Estes passos têm como intuito normalizar todos os gestos desenhados de forma a otimizar C, facilitando a correspondência dos pontos e alinhamento com T_i, como se pode observar na Figura 38 [Wobbrock et al., 2007b].

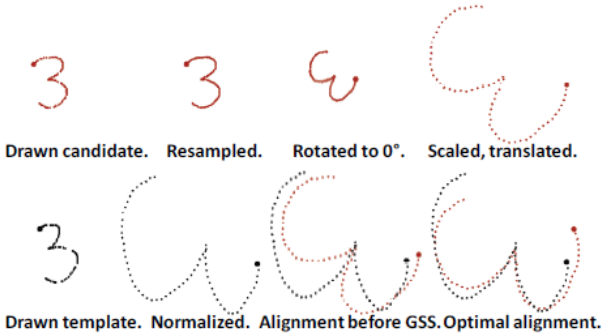


Figura 38 - Descrição dos passos de correspondência e alinhamento de um conjunto de pontos efetuado pelo $1 [Anthony and Wobbrock, 2007b].

5.3 Reconhecedor $N

O $N baseia o seu funcionamento na mesma estratégia definida pelo seu predecessor $1, realizando um conjunto de operações com base em geometria simples e trigonometria. Contudo, a grande melhoria encontra-se na versatilidade e capacidade de reconhecimento de múltiplos traços, aliando-lhe um ótimo desempenho, como pode ser constatado em [Anthony and Wobbrock, 2007a]. A capacidade de generalização deste algoritmo permite determinar no conjunto de amostras existentes, a amostra que mais se assemelha, com base na quantificação e atribuição de pontuações.

Os processos de otimização usados por este algoritmo consideram aspetos essenciais à minimização do tempo de análise e reconhecimento, bem como redução dos recursos computacionais associados ao processo de reconhecimento. Para isso, são considerados aspetos como os ângulos iniciais formados pelo conjunto de amostras e o número de traços, apenas comparando os que têm o mesmo número de traços, permitindo com isto poupar 79.1% de comparações no primeiro parâmetro e 89.5% no segundo, assim como aperfeiçoar

[19] Técnica que consiste em encontrar o limite máximo ou mínimo recorrendo a uma função que analisa a distribuição dos valores dentro desses limites.

a precisão de reconhecimento em 1.3% e 1.7% respetivamente, [Anthony and Wobbrock, 2007b].

A ordem de desenho de um carácter pode-se processar de forma aleatória. Contudo, para cada carácter, existe um número limitado de caminhos permutáveis para desenhar. Este algoritmo considera um conjunto de traços como se se tratasse de um único traço, unindo com um traço adicional o fim do primeiro com o início do seguinte. Posteriormente são realizadas as várias permutações possíveis dos caminhos, e executada a permutação de todas as combinações possíveis para os caminhos de desenho, fazendo assim que o algoritmo seja mais flexível a variações no desenho (Figura 39).

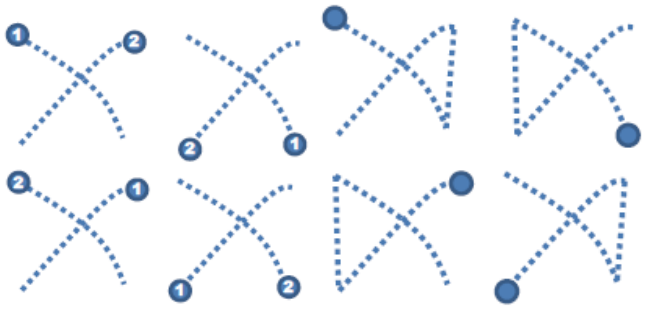


Figura 39 - Possíveis combinações, no desenho de 'x'. Na imagem à esquerda estão representadas quatros diferentes ordens de desenho. À direita vê-se o conjunto de dois traços unidos pelas extremidades de cada traço e convertidos num único traço.

Contrariamente ao $1, que apenas dispunha de uma invariância de rotação total do gesto, o $N oferece invariância limitada. Esta limitação da invariância permite distinguir símbolos semelhantes, embora definidos por orientações distintas e, consequentemente, sentidos diferentes. Por exemplo, as letras 'W' e 'M'. Outra valia que caracteriza o $N é a sua capacidade de distinguir gestos com uma e duas dimensões. A classificação executada pelo $N passa por uma distinção automática entre os dois tipos, sendo que a classificação no caso do 2D é com base nos dois limites (x, y) da caixa delimitadora do conjunto de pontos, e redimensiona os gestos não uniformemente. No caso do 1D, o redimensionamento é uniforme.

Para finalizar, são consideradas ainda duas questões, abordadas por Jacob O.Wobbrock em [Anthony and Wobbrock, 2007b] e que se prendem com a articulação dos gestos realizados e o efeito do número de amostras no rigor do reconhecimento. Relativamente à velocidade de articulação em que os gestos se processam, um aumento da velocidade não implica um aumento da taxa de erros no processo de reconhecimento, visto a distribuição dos pontos variar com o incremento do processo de reamostragem. O aumento do número de amostras melhora consideravelmente a precisão do reconhecimento.

5.4 Classificador conexionista

O conceito de classificador conexionista surgiu de uma conferência realizada no campus de Dartmouth College, da qual partiram as linhas orientadoras deste classificador. Um classificador conexionista tenta simular as intercomunicações existentes entre os neurónios existentes num sistema neuronal biológico [Kolman and Margaliot, 2009].

5.4.1 Redes neuronais

A abordagem realizada às redes neuronais parte de um exemplo implementado e descrito por Mike O'Neill no seu artigo [O'Neill, 2006], que teve por base artigos de dois investigadores na área LeCun e Yann LeCun, a partir dos quais conseguiu elaborar um reconhecedor de caracteres manuscritos usando redes neuronais.

As redes neuronais abrangem um vasto leque de funcionalidades, de forma que são generalizadas. Contudo, a rede neuronal usada no reconhecimento de caracteres é constituída por cinco camadas.

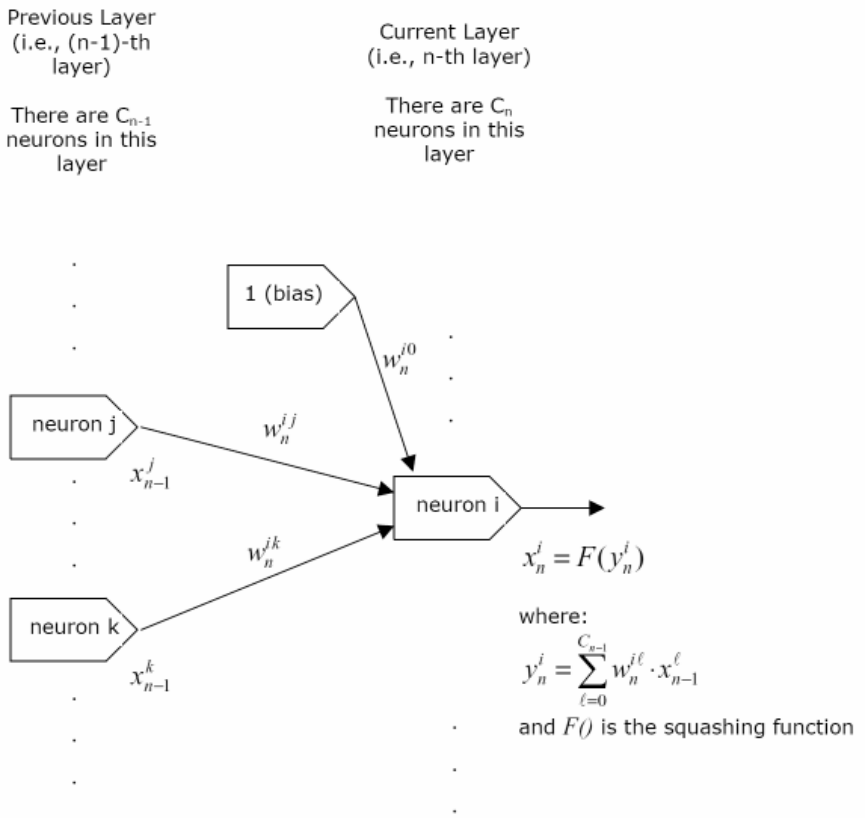


Figura 40 - Passagem dos dados de saída dos neurónios j e k mais o valor bias.

Uma rede neuronal é composta por camadas, sendo estas constituídas por múltiplos neurónios que se alimentam dos dados de saída x dos antecessores, como pode ser

observado na Figura 40. O neurónio posterior, ao receber os x relativos aos vários neurónios antecessores, calculados a partir do valor de entrada nesses neurónios anteriores e aplicando-lhes a fórmula da propagação para a dianteira (Equação 15), na qual i representa o número do neurónio e n a camada; w, por sua vez, representa o peso da saída x de um neurónio e l a entrada do neurónio. Com a propagação dianteira permite que os neurónios posteriores apliquem um peso a cada um dos valores calculados e transmitidos. Por fim, o neurónio posterior executa sobre os pesos, juntamente com o valor da tendência $(bias)$ igual a 1, uma função de activação $F()$ que deve ser a mais simétrica possível, como o exemplo da função hiperbólica mencionada no artigo de Mike O'Neill, obtendo com isso, os valores de saída ou de ativação y num intervalo de -1 a $+1$.

$$x_n^i = F(y_n^i) = F(\sum_{l=0}^{C_{n=1}} w_n^{il} \cdot x_{n-1}^l) \quad \text{Equação 15}$$

Nas redes neuronais, a determinação do erro na saída de uma camada sucede segundo um processo iterativo de retropropagação, partindo do último neurónio para o primeiro. Assim, conhecendo o erro na saída de cada neurónio, é possível calcular as mudanças nos pesos de maneira a corrigir o erro e, consequentemente, reduzi-lo.

Para concluir, não podia deixar de se considerar a aprendizagem inerente a este algoritmo, que demonstra melhorar a precisão do reconhecimento de acordo com os treinos realizados. Isto acontece devido à capacidade de generalização do algoritmo, que vai sendo apurada através da análise de um conjunto de dados para aprendizagem, sujeitando-os a técnicas que visam otimizar a generalização (por exemplo: a ligeira distorção dos padrões).

5.5 Conclusão

A análise dos diferentes tipos de algoritmos de reconhecimento de padrões ajudou na compreensão do modo como os dados de entrada são tratados pelos vários algoritmos, assim como a forma como estes são processados. Dos algoritmos abordados, foram selecionados e avaliados os algoritmos KDA, SVM e $N que usaram como base um conjunto de amostras de caracteres, aplicadas para aprendizagem do algoritmo e que posteriormente serviriam para identificar num conjunto de caracteres constante aos três reconhecedores, por forma a avaliar o grau de precisão de cada reconhecedor. Destes foi optado pelo algoritmo estatístico supervisionado $N, pela à sua simplicidade de implementação baseada em trigonometria simples.

6 Protótipo desenvolvido

Tendo em consideração as tecnologias descritas nas secções anteriores e não perdendo de vista o objetivo inicial de desenvolver uma aplicação NUI para apoio ao processo de ensino e de aprendizagem para crianças do ensino básico, foi criado um protótipo que utiliza o dispositivo Kinect, a biblioteca de desenvolvimento NUI OpenNI/NITE e o reconhecedor de padrões $N. Com isto, são definidas posturas específicas de interação com a aplicação e planos de funcionamento apropriados às diferentes ações que se pretendem realizar com o braço de desenho. O utilizador deverá desenhar as letras pedidas, sendo estes desenhos posteriormente analisados pelo reconhecedor e apresentando os resultados das identificações após concluir todas as tentativas [Leitão et al., 2012][20].

6.1 Interface com o utilizador

A aplicação desenvolvida usa como metáfora organizacional o desenho de símbolos na areia da praia. Esta aplicação está dividida em sete grupos: ecrã introdutório, deteção das articulações, configuração do painel de escrita, desenho de letras, reconhecimento de letras, gravação do desenho de letras e apresentação dos resultados das identificações.

O ecrã inicial (Figura 41) funciona como um cenário introdutório onde são indicados a instituição de ensino e o autor do trabalho realizado. A informação é apresentada em forma de texto, sendo mostrada através de um efeito de *fade in* ao longo de 10 segundos, até que o texto esteja totalmente visível.

[20] Ver apêndice.

Figura 41 - Ecrã introdutório.

O segundo grupo, da deteção das articulações, destina-se à localização e rastreio do utilizador. Para isso, este deverá levantar os dois braços e baloiçá-los alternadamente para a esquerda e para a direita, até que o sistema o detete e reconheça as articulações do corpo. As articulações obtidas são posteriormente representadas através de esferas ligadas por intermédio de segmentos de linhas retas, representando o esqueleto do utilizador e apresentando-o no cenário inicial da praia (Figura 42).

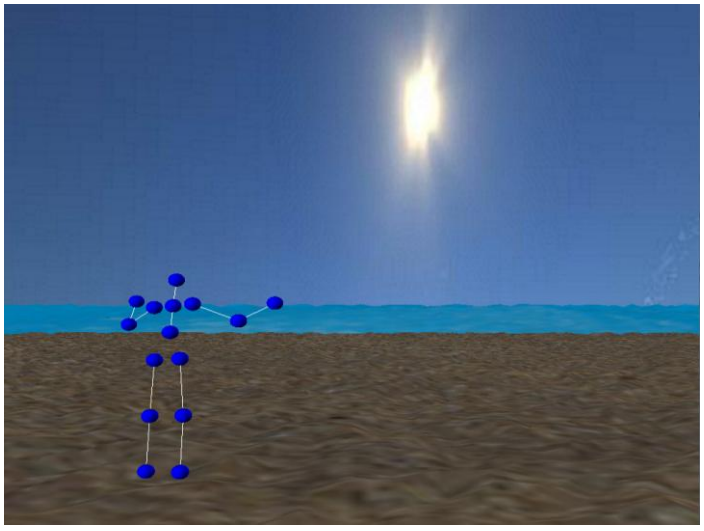


Figura 42 - Ecrã de localização das articulações do utilizador.

Depois de localizar o corpo, e ainda no cenário inicial da praia, segue-se a operação de configuração do painel de escrita. Aqui, o utilizador deve estabelecer a distância entre si e o plano de escrita, movendo a mão direita para o lado direito, aproximadamente à mesma altura do ombro, e de seguida baixando a mão esquerda abaixo do nível da anca esquerda. Após definida a distância do utilizador ao plano de escrita, aquele deve indicar qual das mãos

pretende usar. Assim, quando aparece o ecrã ilustrado na Figura 43, deverá projetar para a frente a mão pretendida.

Figura 43 - Ecrã de indicação ao utilizador, destinado à seleção da mão de desenho que pretende usar.

Depois de determinar os parâmetros inerentes à configuração do painel de desenho, é iniciada uma nova fase (Figura 44). Nesta são concedidas cinco tentativas, nas quais o utilizador deverá acertar em cada uma das letras pedidas. Estas são geradas aleatoriamente de acordo com as 26 letras do alfabeto. O utilizador poderá movimentar o cursor (uma representação de um galho de uma árvore), movendo a mão de desenho ao longo de um plano situado mais próximo do seu corpo, ou então movimentar o cursor ao mesmo tempo que vai adicionando pontos ao ecrã, tendo a mão selecionada de atravessar o plano mais afastado. Terminado o desenho, estão disponíveis duas opções: a de limpar os pontos adicionados ou a de identificar o símbolo desenhado.

Figura 44 - Ecrã de desenho.

Para identificar o conjunto de pontos que constituem o símbolo desenhado, surge um novo cenário para o grupo do reconhecimento de letras, onde é apresentado o resultado da identificação do símbolo. Aqui, estão unicamente disponíveis as opções para guardar ou não o referido símbolo (Figura 45).

Figura 45 - Ecrã de exibição do resultado do reconhecimento. Opção “Não” para retornar ao ecrã anterior. Opção “Sim” para selecionar uma letra a atribuir ao desenho.

Caso não se pretenda guardar, dever-se-á selecionar a opção ‘Não’, sendo automaticamente direcionado para o grupo de desenho de letras (o ecrã anterior, representado na Figura 44). Caso contrário, se selecionar a opção para guardar o desenho, é então que surge um novo cenário (Figura 46), apresentando as 26 letras do alfabeto possíveis de associar ao conjunto de pontos do desenho que se pretende guardar. Para selecionar a letra pretendida, o utilizador deverá projetar a mão de desenho até ao plano mais afastado, escolher a letra e, por fim, acionar o botão para guardar. Esta adição de um novo conjunto de pontos ao conjunto de amostras existentes enriquece o processo de reconhecimento e generalização de futuros desenhos.

Figura 46 - Ecrã de seleção da letra a atribuir á letra desenhada.

Por fim, após terminarem as cinco tentativas disponíveis para o utilizador responder às letras geradas pela aplicação, são apresentados os resultados da identificação de cada símbolo pelo algoritmo de reconhecimento de padrões. As letras corretamente identificadas são assinaladas com um "visto" verde. Os erros são marcados com uma "cruz" vermelha (Figura 47).

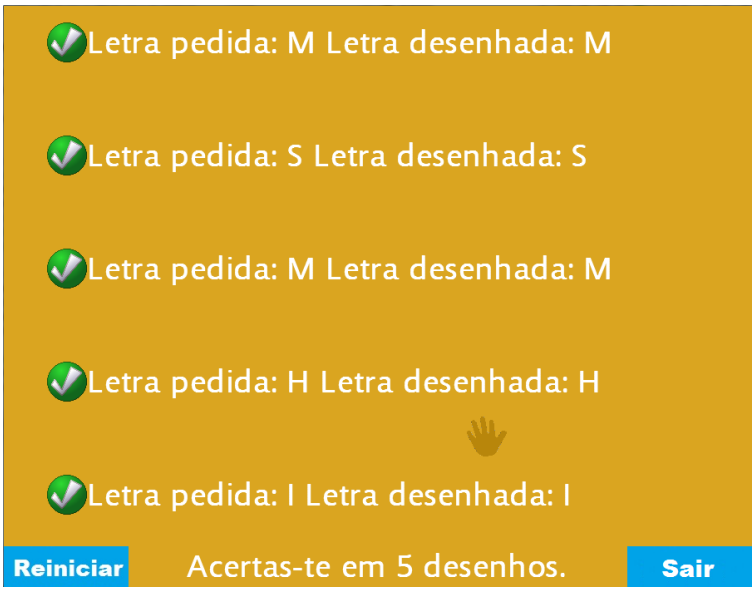


Figura 47 - Resultado dos reconhecimentos dos símbolos desenhados.

6.2 Componente funcional

Nesta componente, começou-se por implementar os algoritmos KDA, SVM e $N no protótipo da aplicação, por forma a averiguar o grau de eficiência em cada um destes, com base no mesmo número de testes aplicados aos três.

O reconhecimento dos desenhos é descrito através de dois processos distintos para os três diferentes algoritmos aplicados. Nos casos do KDA e do SVM, ambos são sujeitos a três fases para obtenção do resultado do reconhecimento. Começa-se pelo carregamento de um conjunto de 260 amostras de letras limiarizadas em matrizes de 32 por 32. Ambos os algoritmos são configuráveis para encontrar uma gama de valores que garantam uma melhor eficiência [Lin et al., 2008]. Os valores aplicados aos parâmetros foram os estipulados por César de Souza em [Souza, 2012f] e [Souza, 2010a].

Depois do carregamento das matrizes das amostras e das respetivas identificações, o KDA constrói o objeto `KernelDiscriminantAnalysis`, aplicando-lhe as matrizes de entrada, as suas identificações e o tipo de função Kernel a usar. Com isto, o conjunto de dados é analisado, sendo-lhe aplicada uma função gaussiana definida por três parâmetros. O valor gaussiano `kernel = new Gaussian(6.2200)` atribuído à função, representa o valor padrão a usar na distribuição gaussiana, valor este ajustável, permitindo otimizar a busca no conjunto de modelos existentes. Por sua vez, o regularizador `Regularization` estipula valores únicos na solução durante a análise dos modelos. Já o limar `Threshold` funciona como limite de decisão. No caso de um discriminante não usado, deve ou não ser descartado pela aplicação caso se encontre abaixo desse limiar após a análise [Kardi Teknomo's, 2012].

Tal como o algoritmo anterior, o SVM também usa uma função gaussiana `kernel = new Gaussian(6.2200)` com vista a otimizar a busca no conjunto de modelos existentes. Este algoritmo começa por criar as múltiplas classes para serem usadas pela função kernel, definida através de três parâmetros: os dados de entrada, a função kernel a aplicar da máquina de vetor suporte, e o número de classes a usar no processo de classificação. O número de classes deve estar diretamente relacionando com a heterogeneidade dos dados de entrada usados, neste caso as 26 letras do alfabeto.

Antes de computar as amostras usando os vetores de suporte para a aprendizagem, definem-se os parâmetros `Complexity`, `Epsilon` e a `Tolerance` do algoritmo SMO. Um aumento do valor da complexidade implica a criação de um modelo mais preciso e de uma perda na capacidade de generalização das amostras. O valor de `Epsilon` representa a largura da margem de regressão. Assim, quanto maior for este valor, maior será esta margem e, consequentemente, menor o número de vetores de suporte considerados para classificação. O contrário acontece quando esta margem é menor o que proporciona uma classificação mais abrangente. A tolerância de convergência `Tolerance` aplicada é limitada entre valores de 0.01 a 0.001 [Andrew, 2011]. Por fim, é obtida a classificação dos dados de entrada do utilizador e retornado o índice da letra como resultado.

A arquitetura inicial da aplicação, representada na Figura 49, descreve um sistema constituído por um módulo ScreenManager, e cinco restantes subgrupos: Skeleton, Scenes, Draw, Camera e Components. O grupo ScreenManager é responsável por gerir os diferentes cenários contidos no grupo Scenes representados no diagrama de classes da Figura 48, permitindo que, ao longo da execução da aplicação, seja possível navegar ao longo desses ecrãs.

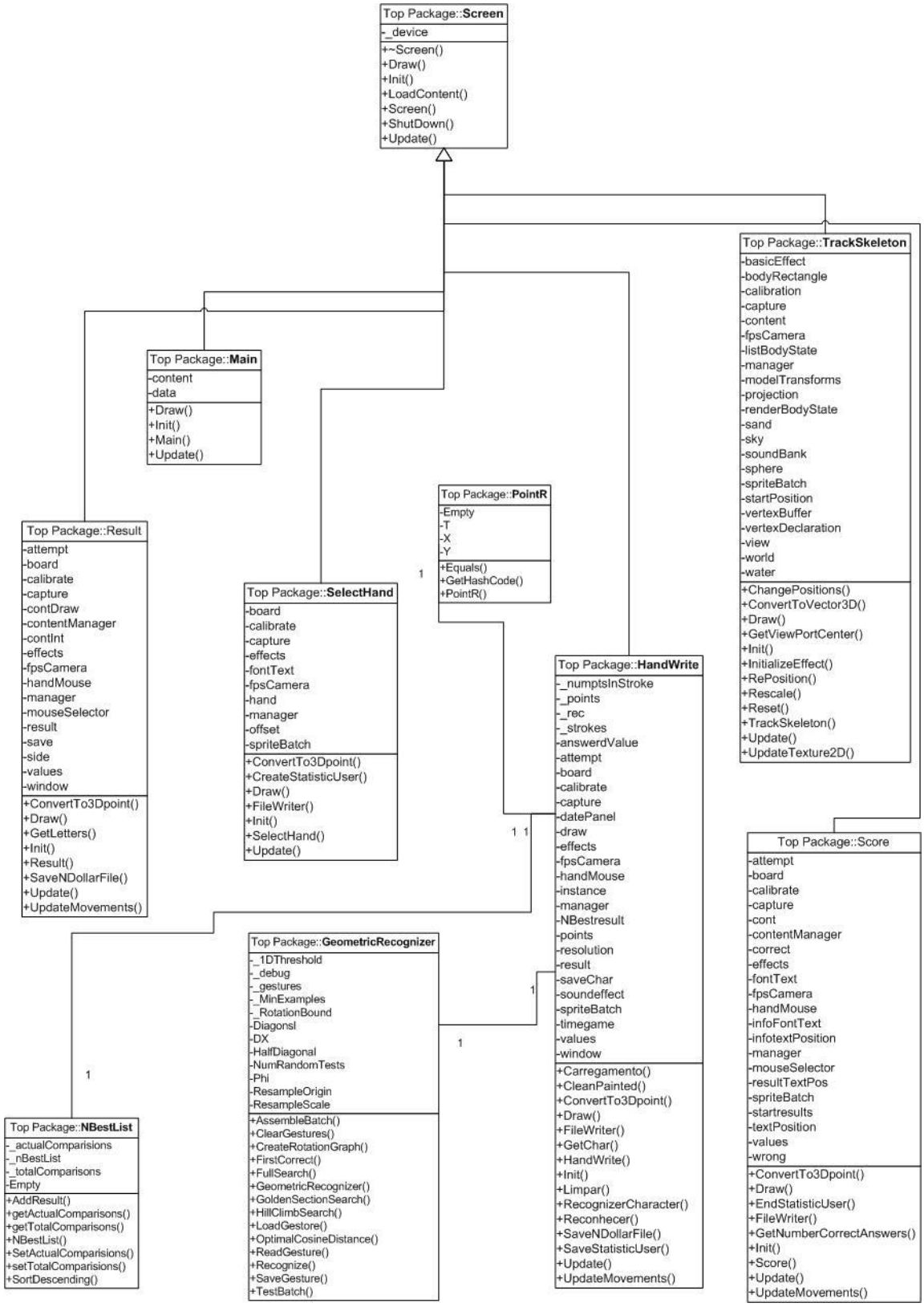


Figura 48 - Diagrama de classes da aplicação.

O módulo Skeleton contém as classes BonesSpheres, Calibration e CaptureJoints, através dos quais é possível detetar e representar as 15 articulações do corpo num ambiente tridimensional através de esferas, assim como definir e descobrir a postura na qual a distância do utilizador ao plano é obtida. Os planos de interação são estabelecidos de acordo com o valor da medição no módulo Skeleton da distância do ombro à mão direita. Com base nesse valor, o módulo Draw cria três limites: o primeiro, de 0% a 25% que corresponde ao segmento de inatividade de interação; um segundo segmento, de 25% a 75%, destina-se à movimentação do cursor; e por último de 75% a 100%, para escrita e movimentação do cursor pelo ecrã. O módulo Camera, como o nome indica, representa o ponto de visão de onde surge a imagem obtida e exibida do ambiente virtual. Por último, o módulo Components, contém classes mais gerais que servem para auxiliar o funcionamento da aplicação. Por exemplo, a classe Buttons, que se destina à criação de botões de seleção [Jacobson, 2012] nos diferentes cenários; a classe Attempts, usada para indicar o número de tentativas; SaveCharacter é usada para guardar o conjunto de pontos das letras desenhadas em formato XML junto às restantes amostras; ParseRecogResult serve para obter a letra selecionada no painel de escolha da letra com a qual o utilizador pretende guardar a amostra da letra desenhada.

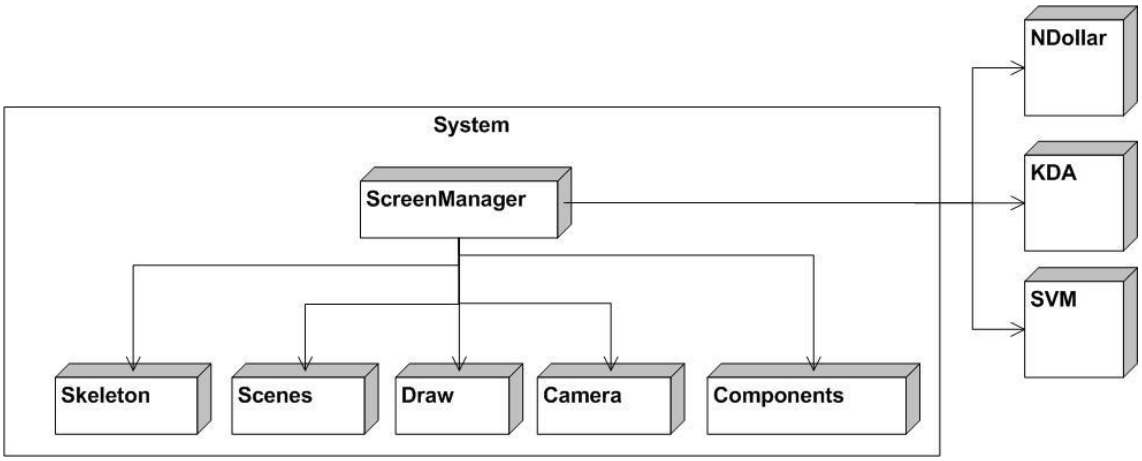


Figura 49 - Arquitetura da aplicação.

6.3 Desenvolvimento da aplicação

O projeto iniciou-se com a recolha das coordenadas de cada uma das articulações capturadas pelo Kinect. Este primeiro teste usou a plataforma OpenNI, juntamente com a biblioteca NITE, e realizou-se recorrendo a duas aplicações do tipo de linhas de comandos, funcionando segundo um modelo computacional dividido cliente/servidor, comunicando estas duas aplicações entre si. Neste modelo de computação distribuída, a aplicação servidor é responsável por responder a pedidos enviados pela aplicação cliente. Usando a aplicação servidor OSCeleton [OSCeleton, 2011], esta obtém os dados das articulações do corpo humano recolhidos pelo OpenNI/NITE, capturando as coordenadas tridimensionais de cada uma dessas articulações e enviando-as através de mensagens, via protocolo OSC, para a

aplicação cliente [Bespoke Software, 2011]. Por sua vez, a aplicação cliente representa os valores das coordenadas das articulações recebidas (Figura 50). A aplicação servidora, depois de iniciada para realizar corretamente a captura, necessita que lhe sejam definidos parâmetros de execução tais como $-mx$ e $-my$ para multiplicar as coordenadas recolhidas com diferentes tipos de resoluções intrínsecos ao dispositivo e, por último, o parâmetro $-ox$, que ajusta o tamanho do esqueleto. Uma vez configurados estes parâmetros, é dado início à transmissão das mensagens pela aplicação OSCeleton.

$$OSCeleton.exe - mx640 - my480 - ox160$$

Figura 50 - Captura das articulações do corpo humano. À direita encontra-se a aplicação servidora OSCeleton; à esquerda a aplicação cliente, que recebe os valores transmitidos pelo OSCeleton e representa-os através de esferas [SenseBloom, 2011].

Inicialmente a aplicação começa por detetar 15 articulações do corpo do utilizador (Figura 51 (a)) através de pontos azuis, atualizando constantemente as coordenadas tridimensionais das posições desses pontos. Para a aplicação obter as coordenadas dessas articulações, o utilizador deve começar por indicar a sua presença perante o dispositivo Microsoft Kinect, elevando os braços acima da cintura e acenando continuamente com o corpo para a esquerda e para a direita, (Figura 51 (b)).

(a) (b)

Figura 51 - (a) Conjunto de pontos identificados e capturados pelo OpenNI/NITE. (b) Detecção da presença do utilizador, acenando com os braços acima da cintura.

O plano de escrita consiste, na realidade, num plano de dupla profundidade, estabelecido com base na medição da distância medida da mão direita ao ombro direito do utilizador (Figura 52).

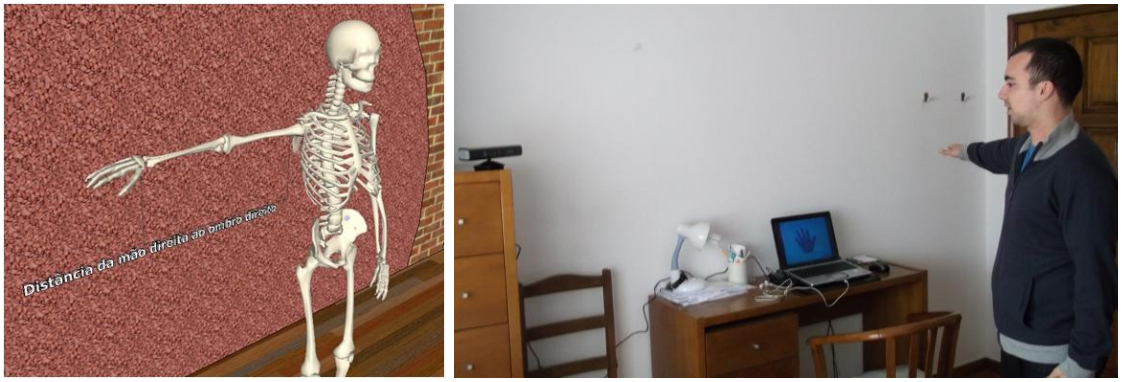


Figura 52 - Postura definida para a determinação do duplo plano de interacção.

As áreas de interação no ar foram pensadas como um conjunto de três segmentos, definidos em função do valor obtido da medição referida anteriormente. Esse valor é segmentado em três (Figura 53), ficando assim o segmento que vai dos 0% aos 25% a tratar-se de um segmento de ócio. Assim, sempre que a mão de escrita se encontrar dentro deste segmento, não será desencadeada qualquer ação.

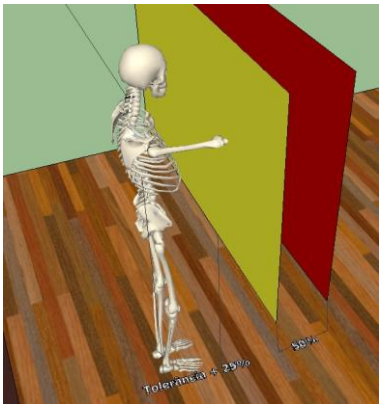


Figura 53 - Plano de escrita de duplo limite. O segmento que vai do plano amarelo ao plano encarnado (50%) destina-se à movimentação e seleção do cursor. Do plano encarnado para trás destina-se ao desenho de linhas e pintura da tela.

Usando as posições obtidas pelo processo de captura das articulações, cada uma destas é representada através de esferas azuis. Porém, devido à escala na qual as coordenadas são obtidas não ser a mais adequada, é necessário proceder à conversão usando a função `Rescale(Point3D position)`, que dimensiona cada posição para uma escala 100 vezes inferior, podendo assim representar corretamente cada uma das esferas no ecrã.

As esferas representadas são sujeitas a um conjunto de transformações rígidas das quais se destaca uma translação. Possibilitando a constante modificação das posições das esferas, transladando-as para as novas posições adquiridas sem que, para tal resultem alterações nas formas ou dimensões destas [Jared, 2011].

As transformações rígidas incluem um conjunto de operações. Contudo, a única aplicável foi a de translação, pois o dispositivo Kinect não é capaz de detetar a rotação dos membros.

Numa fase inicial, a definição do funcionamento do plano de desenho passou pelo estabelecimento de apenas dois segmentos, um para interação e outro para interação e escrita. Na Figura 54, o utilizador dispõe de um segmento que vai desde a sua posição até à profundidade equivalente ao comprimento do seu braço direito. Nestes segmentos, o utilizador usa o primeiro segmento para movimentar e selecionar opções disponíveis ao longo da navegação na aplicação. O último plano, além de permitir a interação, oferece ainda a possibilidade de desenhar sobre a tela do monitor, adicionando-lhe pontos.

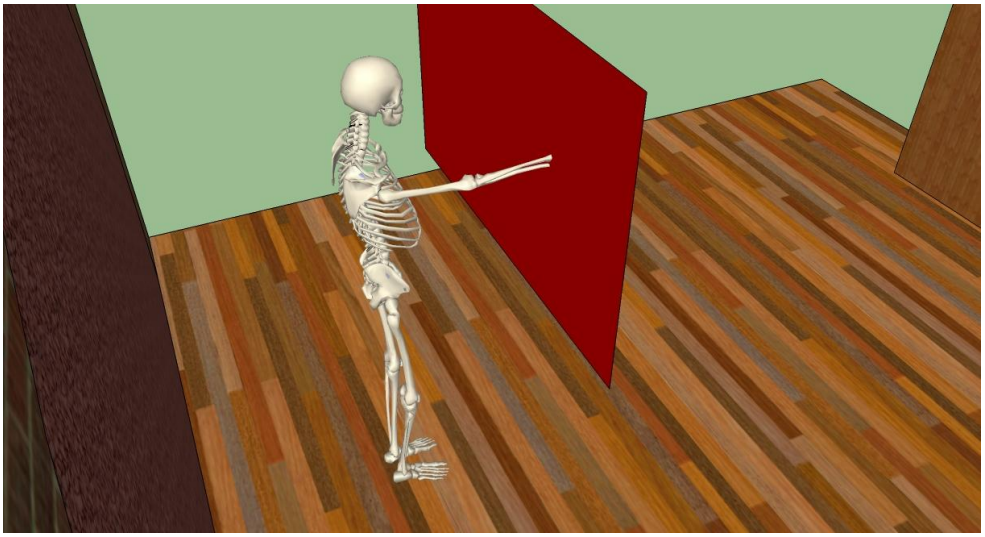


Figura 54 - Determinação de dois segmentos de interação. Espaço que antecede o plano assinalado a encarnado, destinado à interação e movimentação. Espaço que precede o plano encarnado, destinando à interação e desenho.

O plano de escrita no ar foi posteriormente pensado como um conjunto de dois limites usando dois segmentos, definidos segundo o valor obtido do comprimento do braço direito (Figura 53). O valor do comprimento entre a mão e o ombro direito é segmentado em dois. O primeiro segmento é destinado à movimentação do cursor ao longo da tela para interação com os componentes. O segundo segmento destina-se exclusivamente à pintura da tela ao longo do movimento reproduzido no ar e transposto para a mesma. O segmento que vai dos 0% aos 25% não tem quaisquer efeitos na interação com a aplicação. Assim, sempre que a mão de escrita selecionada se encontre dentro deste não realizará qualquer ação. O segmento que vai dos 25% aos 75% corresponde ao maior segmento, e é através deste que o utilizador deverá interagir maioritariamente por se revelar essencial na movimentação e interação com a aplicação. Sempre que o braço seja movimentado dentro destes limites as coordenadas bidimensionais (x, y) da mão serão transpostas para a tela, movimentando o cursor também segundo as coordenadas bidimensionais correspondentes.

O último segmento vai dos 75% aos 100% e funciona à semelhança do anterior, com a transposição das coordenadas da mão para o cursor na tela. Contudo, é apenas necessário transpor este limite apenas quando do desenho das linhas (Figura 55).

Figura 55 - Utilizador com a mão projetada prestes a ultrapassar o limite estipulado para o desenho.

De forma a aperfeiçoar o modo de interação, foi implementado e testado um novo método (Figura 56). Com base nos três segmentos anteriormente definidos, usava-se a localização da mão de desenho, guardando a função do segmento anterior e transpô-la para o segmento atual. Dependendo da posição da mão, caso esta ultrapasse os 25%, fazendo com que não desenhe sobre a tela; ou caso ultrapasse os 75%, ativando a função de escrita. Com isto, o segmento que equivale aos 50% 'mexe/não mexe' estabelecido entre os segmentos de desenho com a tela 'ESCREVE' e o segmento de ócio 'MEXE/NÃO MEXE', varia consoante a função do segmento anterior. Assim, sempre que a mão se encontre no segmento 'ESCREVE' destinado à escrita e posteriormente siga para o segmento intermédio a operação neste será a mesma que no segmento anterior. O mesmo acontece quando a mão se encontra no segmento de desenho e recue para o segmento intermédio.

Figura 56 - Ilustração do método de interação baseado na operação do estado anterior.

Depois de vários testes usando este modo de interação, constatou-se a inviabilidade do mesmo, causada pela dificuldade acrescida por parte do utilizador na conclusão dos desenhos das letras. Sempre que o utilizador pretende remover a mão do segmento de desenho, após ter concluído um traço, esta é transportada para o segmento relativo aos 50%, assumindo a

função de desenho passada do segmento anterior. Com isto, o utilizador depara-se com uma dificuldade acrescida, que consiste na remoção da mão dos segmentos de 25% (ESCREVE) e do de 50% (MEXE) para o segmento 'NÃO MEXE' sem adicionar traços indesejados.

Uma das questões iniciais prendia-se com qual o processo ideal para adicionar pontos produzidos durante a fase de desenho. Para esse efeito, foi definida uma matriz constituída por 32x32 quadrados brancos. Os quadrados, numa fase inicial, estão representados a branco, e cada vez que a mão do utilizador ultrapassasse o segmento definido para desenho e o cursor se encontrasse dentro da região a branco (Figura 57), estes comutavam a cor de cada quadrado para a cor preta segundo as coordenadas bidimensionais do cursor.

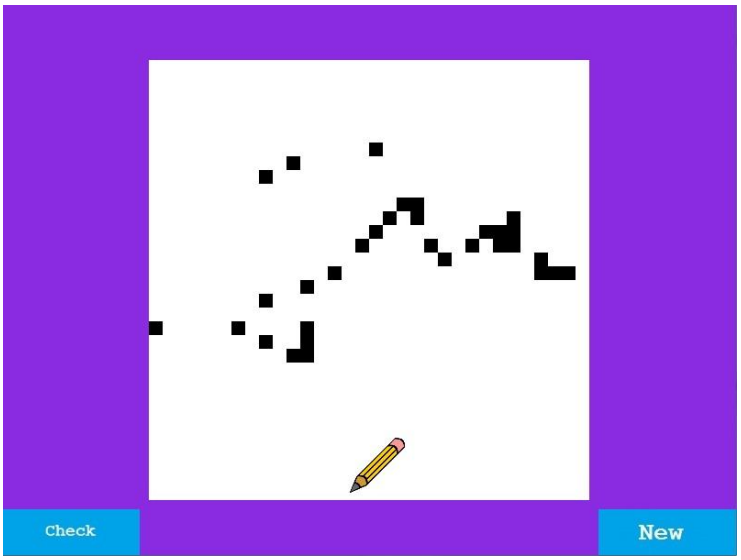


Figura 57 - Interface de desenho utilizando uma matriz 32 por 32 entradas.

Apesar do processo descrito ser funcional, o método de desenho não foi adotado por conter algumas debilidades relativas à interação entre o utilizador e o plano de desenho, dificultando a tarefa de adicionar pontos à tela. Outro aspeto considerado foi o facto de este método ser computacionalmente pesado, originado pelas necessárias verificações constantes dos estados dos 1024 quadrados implementados.

Ainda neste processo, aquando da definição da mão de desenho, a mão oposta à mão de desenho é assumida como mão para apagar os píxeis pretos adicionados (Figura 58). Assim, sempre que este passe por cima dos quadrados comutados para a cor preta, estes regressam à cor branca inicial.

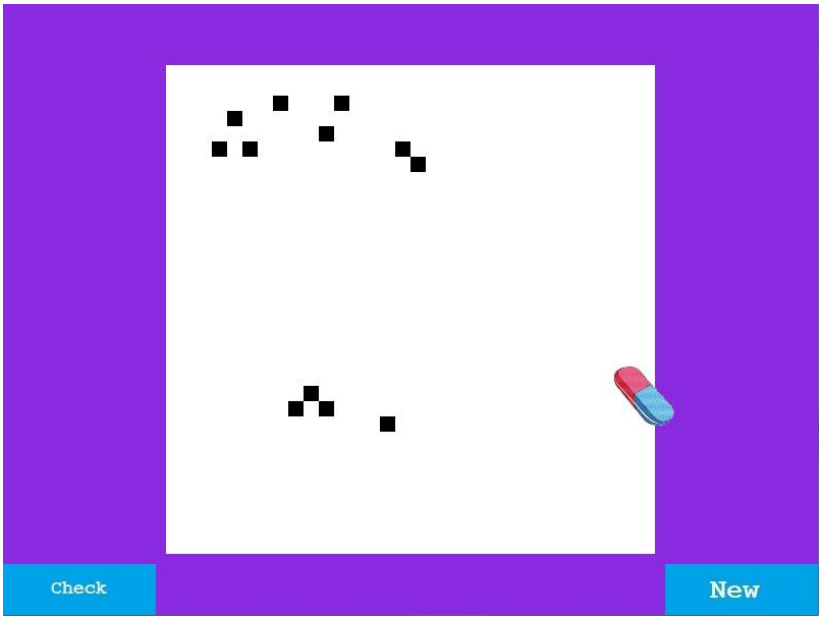


Figura 58 - Mão definida para apagar pontos adicionados ao painel de desenho.

Este processo foi pensado de forma a não limitar a interação exclusivamente a utilizadores dextros, mas também ter em consideração o público sinistro. Assim, o utilizador deverá projetar para a frente a mão que pretende usar, de modo a intercetar o segmento de desenho (75%), como se encontra exemplificado na Figura 55.

Após o desenho estar terminado, é analisado pelo reconhecedor, sendo indicado o resultado da classificação e exibido o padrão que mais se assemelha de entre o conjunto de padrões de aprendizagem usado (Figura 59). Depois do resultado do reconhecimento ser apresentado, ficam disponíveis duas opções, a de voltar ao ecrã anterior para realizar outra das 5 tentativas disponíveis, aproximar as palmas das mão para sair da aplicação, ou então apresentar as opções de gravação do desenho, independentemente do resultado apurado e independentemente de este estar correto ou não. O desenho é adicionado ao conjunto de amostras existentes, melhorando assim o processo de reconhecimento e generalização de futuros desenhos.

Figura 59 - Apresentação ao utilizador da letra reconhecida. Disponibilização das opções para voltar ao ecrã anterior de escrita ou gravar o desenho realizado juntamente com as amostras existentes.

6.3.1 Deteção das articulações

A localização das articulações ocorre no ecrã TrackSkeleton e baseia-se na implementação 'UserTracker.net', contida nos exemplos fornecidos na instalação da plataforma OpenNI. Antes de iniciar a deteção OpenNI, este necessita da configuração de parâmetros de funcionamento definidos num ficheiro XML anexado à aplicação. Esse ficheiro é iniciado pelo elemento base <OpenNI>, a partir do qual são adicionados os restantes elementos de configuração [OpenNI, 2011b]:

`<Licenses>:` neste elemento são definidos dois parâmetros, o `vendor` que representa a entidade que usa a plataforma, e a respetiva licença necessária `key`.

`<Log>:` contém o parâmetro `writeToConsole`. 'True' indica que a sessão pode ser executada em modo consola de comandos. Segue-se o parâmetro `writeToFile`, a 'false', que indica que a sessão não deverá ser escrita para um ficheiro no diretório de trabalho. Outros elementos adicionais a este, e definidos intrinsecamente são: `LogLevel`, com o valor 3, especificando rigor a aplicar na sessão de captura; e `Masks`, a indicar que a máscara se encontra no estado descativado.
`<ProductionNodes>:` este elemento define os nós de configuração, declarando-os pela *tag* `<Node>`. Nas configurações das câmaras RGB e IR, `MapOutputMode` indica a resolução e a frequência de amostragem destes dispositivos especificando o tipo `type` de nó. Por último, o parâmetro `Mirror` adota o valor 'true' para oferecer ao desenvolvedor a permissão para operar com classes do tipo Generators, representadas no código de inicialização para deteção das articulações.

```
<OpenNI>
        <Licenses>
                <License vendor="vendor" key="0KOIk2JeIBYClPWVnMoRKn5cdY4="/>
        </Licenses>
        <Log writeToConsole="true" writeToFile="false">
                <!-- 0 - Verbose, 1 - Info, 2 - Warning, 3 - Error (default) -->
                <LogLevel value="3"/>
```

```
                <Masks>
                        <Mask name="ALL" on="false"/>
                </Masks>
                <Dumps>
                </Dumps>
        </Log>
        <ProductionNodes>
                <!-- Normal Image -->
                <Node type="Image" name="Image1">
                        <Configuration>
                                <MapOutputMode xRes="640" yRes="480" FPS="30"/>
                                <Mirror on="true"/>
                        </Configuration>
                </Node>
                <Node type="Depth" name="Depth1">
                        <Configuration>
                                <MapOutputMode xRes="640" yRes="480" FPS="30"/>
                                <Mirror on="true"/>
                        </Configuration>
                </Node>
        </ProductionNodes>
</OpenNI>
```

Ficheiro de configuração OpenNI.

A captura das várias articulações e a representação das suas posições é feita através de esferas, as quais sofrem uma constante atualização das posições tridimensionais através dos dados capturados pelo dispositivo Kinect. Posteriormente as esferas são dimensionadas para uma escala 1:100, mostrando um modelo que reflete todos os gestos realizados pelo utilizador. Representando as articulações com esferas, resta tornar a representação mais realista. Para isso foram usadas primitivas tridimensionais para unir corretamente as articulações por intermédio de linhas (`PrimitiveType.LineList`) definidas entre dois pontos [Microsoft MSDN, 2012c]. A matriz de transformações aplicada às esferas é também usada pelos pontos iniciais e finais no desenho das primitivas, garantindo o sincronismo na translação entre esferas e linhas.

6.3.2 Configuração do painel de desenho

O painel de desenho, referido anteriormente (seção 6.3), é estabelecido através do valor obtido na medição entre o ombro e a mão direita e o posicionamento da mão esquerda abaixo da anca esquerda. Esta implementação usa um conjunto de verificações das posições dos membros. As posições deles são verificadas nas funções `CheckUnderHip()` e `CheckShoulderLevel()`. Caso o valor retornado por estas verificações seja positivo, a medição é executada recorrendo à função `GetRightHandToRightShoulderDistance()`. Um fator a considerar no posicionamento do braço direito ao nível do respetivo ombro é a necessidade da definição de uma margem de tolerância (`tolerance = 20.0f`) entre estes, de forma a evitar que o utilizador tenha de nivelar o braço exatamente ao nível do ombro (Figura 60). Este aspeto origina que os distanciamentos ao plano sejam diferentes consoante o ângulo de abertura do braço for menor. Depois de vários testes, a definição do plano de escrita mais próximo ou afastado conforme o ângulo de abertura do braço revelou-se um aspeto irrelevante devido ao desfasamento ser mínimo.

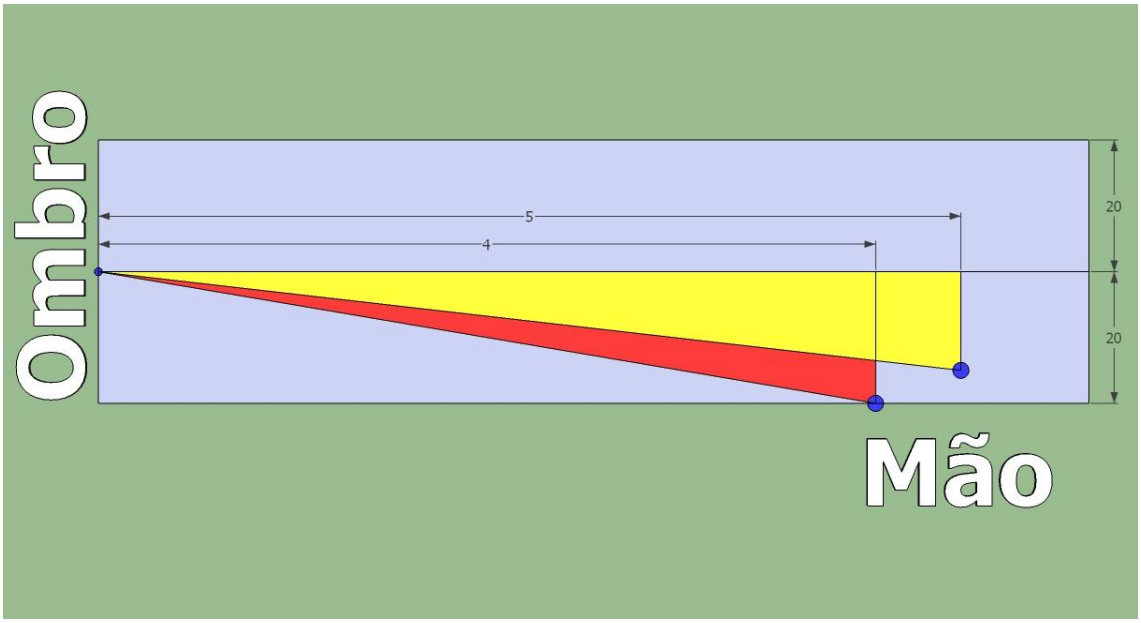


Figura 60 - Ilustração do desfasamento das medições entre o ombro e a mão dentro da margem de tolerância, com diferentes amplitudes.

Do centro do tronco do corpo, mais especificamente no `Torso`, parte um plano retangular com uma distância em *z* equivalente ao comprimento medido anteriormente entre o ombro e a mão, compensado com um deslocamento adicional `offset`. Esta compensação deve-se ao facto de o ponto relativo ao `Torso` se encontrar embutido no interior do corpo, tendo de ser compensado no eixo *z* (Figura 61).

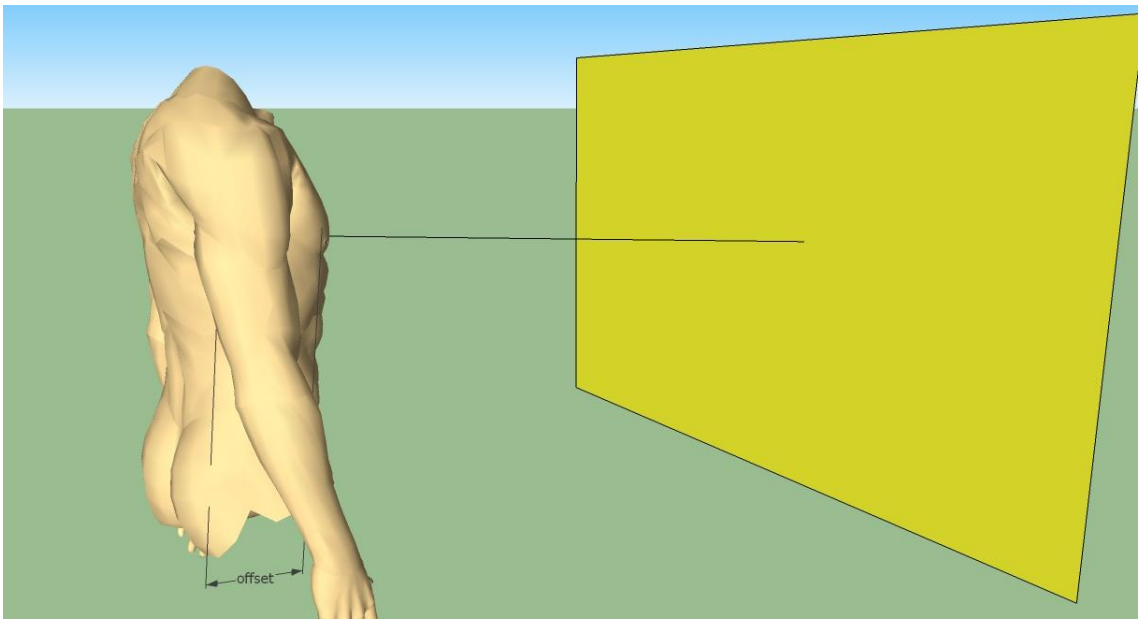


Figura 61 - Ilustração da projeção do plano de escrita, tendo como base o centro do tronco, e considerando o deslocamento deste ponto no interior do corpo.

A operação seguinte consiste na definição da mão pretendida para desenhar. É realizado um conjunto de verificações para determinar se cada uma das mãos está dentro dos limites (x, y) do plano e se o intercepta também no eixo z. Caso estas condições se verifiquem, é passada para os ecrãs seguintes a indicação de qual das mãos intercetou o plano, a qual será futuramente usada ao longo da aplicação.

Um dos testes realizados neste plano passou pela definição de um espaço bidimensional, à semelhança da tela do monitor, onde a movimentação do braço na frente do utilizador é feita sobre um espaço bidirecional (x, y), transpondo diretamente as coordenadas da mão para as coordenadas do cursor. Assim, de cada vez que o utilizador realiza um gesto, o cursor move-se ao longo da tela de acordo com o posicionamento obtido para a sua mão de escrita. Arrastando um pouco mais o braço para a frente, isto é, é alcançado o plano de escrita, iniciando-se a operação de desenho.

6.3.3 Representação da água do mar

Pretende-se representar a água realisticamente, mostrando a ondulação da água do mar. Decidiu-se uma malha de triângulos e atribuindo aos seus vértices movimentos ondulatórios [Fuchs, 2010].

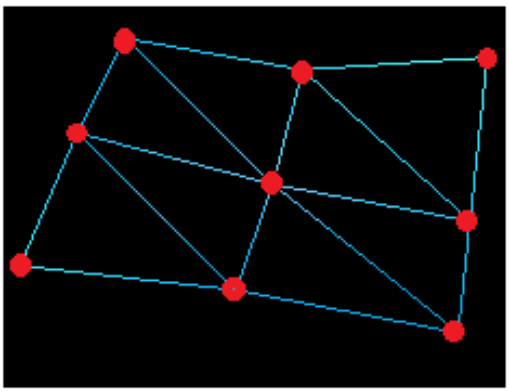


Figura 62 - Interligação dos triângulos da água.

A superfície da água é constituída por um conjunto de quadrados, que por sua vez, são compostos por triângulos, como pode-se observar na Figura 62. Cada um dos quadrados inclui nove vértices, que se vão movendo aleatoriamente entre dois limites: um superior e um inferior. Quando um vértice se encontra verticalmente afastado do valor determinado pelos limites, este move-se para cima ou para baixo de acordo com a sua posição atual, até alcançar a coordenada pretendida. Assim que a alcança é gerado um novo valor, para onde se terá novamente de se mover, realizando esta operação continuamente.

6.3.4 Efeitos sonoros

A arquitetura usada pelo XACT baseia-se em quatro tipos de blocos (Figura 63). São eles os ficheiros áudio *Waves* (.wav), que podem simplesmente ser reproduzidos independentemente usando *SoundEffects*, ou então agrupando os vários ficheiros num único, usando a interface XACT. Depois de agrupar as várias *Waves* num único bloco *Wave Bank*, estes ficheiros são passados ao *Sound Bank,* criando um novo bloco que possibilita a reprodução de cada um dos sons depositados como *Sound Bank* ou a reprodução segundo um novo tipo denominado *Cue*.

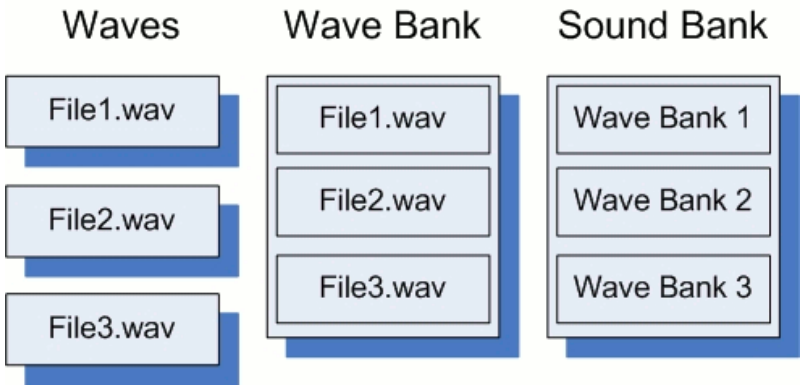


Figura 63 - Arquitectura XACT.

6.3.5 Plano de fundo

A imagem de fundo tem como objetivo tornar a aplicação mais apelativa para as crianças. Além dos componentes já referidos decidiu-se representar o céu e o Sol. À semelhança da representação da superfície da água do mar, neste foram apenas definidos quatro vértices relativos às arestas da imagem a ser usada como plano de fundo.

6.3.6 Desenho de letras

Os desenhos produzidos ao longo da tela consistem num conjunto de pontos adicionados sempre que a mão de desenho entra em contacto com o segmento definido para a escrita. Assim, na seleção da mão de escrita para se realizar um traço é necessário verificar se esta se encontra dentro dos limites (x, y, z) estipulados. Caso esta condição se realize, além de atualizar a posição (x, y) do cursor, adiciona as mesmas coordenadas bidimensionais a uma lista de pontos `_points`. Quando um traço está terminado e a mão é retirada do segmento de desenho, todas as coordenadas acumuladas na lista de pontos `_points` são descarregados para uma lista de traços `_strokes` e a lista de pontos `_points` é esvaziada.

6.3.7 Reconhecimento de letras

Foram implementados três algoritmos distintos de reconhecimento: KDA, SVM e $N. Nas subsecções que se seguem apresentam-se alguns detalhes deste processo.

6.3.7.1 KDA e SVM

Um fator a considerar são as particularidades inerentes às amostras dos algoritmos: o KDA e o SVM usam um tipo de amostras que se baseia na limiarização da imagem, como se pode ver pela representação da letra 'A' através de 0's e 1's. Esta limiarização consistiu em atribuir o valor '1' aos traços, e o valor '0' ao restante ecrã de fundo (Figura 64).

```
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000110000000000000
00000000000000001111000000000000
00000000000000011111000000000000
00000000000000111110000000000000
00000000000001111111000000000000
00000000000011111111000000000000
00000000000011111111000000000000
00000000000111101111100000000000
00000000000111101111100000000000
00000000001111000111100000000000
00000000001111000111100000000000
00000000001111000111100000000000
00000000001111000011100000000000
00000000011100000011100000000000
00000000011111000001111000000000
00000000111111111111111000000000
00000001111111111111111000000000
00000001111000000001111000000000
00000011110000000001111000000000
00000011110000000001111000000000
00000111100000000001111000000000
00000111100000000001111000000000
00000111100000000000111000000000
00000011000000000000111000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000

A
```

Figura 64 - Matriz da letra desenhada, e respetiva identificação pela letra 'A'.

As letras desenhadas são convertidas num ficheiro de imagem Joint Photographic Experts Group (JPEG), de dimensões 400x400. O conjunto de amostras, ao ter sido analisado pela aplicação como pequenos conjuntos de matrizes de 32x32, leva a que o desenho, para ser classificado corretamente, deva também ser classificado como uma matriz 32x32. Efetua-se o redimensionamento da imagem anterior, realizando por fim a sua limiarização, obtendo assim a matriz de '0's e '1's desejada (Figura 65).

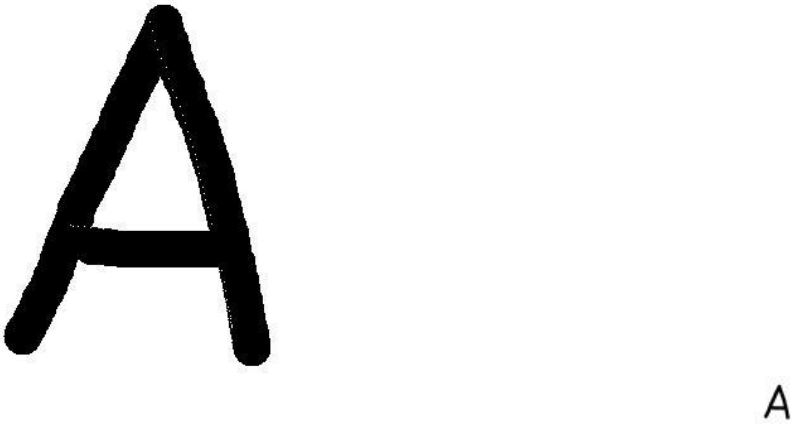


Figura 65 - Desenhos passados da tela do monitor para imagens JPEG. Primeiro passo à esquerda, a imagem é passada diretamente do monitor com 400x400. Na da direita, a imagem anterior é dimensionada para 32x32.

6.3.7.2 $N

O $N, por sua vez, usa um conjunto de amostras definido com base nos pontos recolhidos ao longo da realização dos traços. Na seguinte estrutura XML, estão definidos dois traços, cada um constituído por um conjunto de pontos.

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<Gesture  Name="A_1"  Subject="test"  Speed="test"  NumPts="162"  Milliseconds="3229"
AppName="Recognizer.NDollar" AppVer="4.4.2011.0" Date="quinta-feira, 3 de Maio de 2012"
TimeOfDay="19:35:41">
  <Stroke index="1">
    <Point X="83" Y="215" T="25279806" />
    <Point X="83" Y="212" T="25279821" />
    ...
    <Point X="181" Y="199" T="25280820" />
    <Point X="181" Y="200" T="25280835" />
  </Stroke>
  <Stroke index="2">
    <Point X="95" Y="146" T="25282255" />
    <Point X="96" Y="146" T="25282302" />
    ...
    <Point X="172" Y="147" T="25282957" />
    <Point X="173" Y="147" T="25283035" />
  </Stroke>
</Gesture>
```

Estrutura XML de uma amostra constituída por um conjunto de pontos correspondente a letras do alfabeto.

É aplicado um reconhecedor de padrões ao conjunto de pontos constituídos por coordenadas bidimensionais recolhidas do símbolo desenhado sobre a tela, analisando-os de forma a encontrar o símbolo corresponde ao desenhado efetuado. O reconhecedor $N apoia a sua análise num conjunto de 41 amostras relativas às letras do alfabeto. A razão pela qual o número de amostras usado deve ser superior ao número de letras que constituem esse alfabeto, deve-se ao facto de cada uma das letras poder ser desenhada de diferentes formas,

partindo o traço de diferentes posições e variando o número de traços usado de acordo com a forma como a letra é traçada (Figura 66 e Tabela 1).

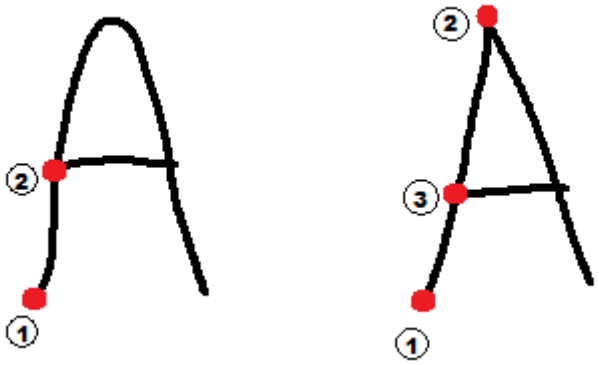


Figura 66 - A letra 'A' pode ser desenhada com 3 traços ou simplesmente usando apenas 2.

Número de amostras para cada letra do alfabeto			
A	2	**N**	2
B	2	**O**	1
C	1	**P**	2
D	2	**Q**	1
E	2	**R**	2
F	2	**S**	1
G	2	**T**	1
H	1	**U**	1
I	1	**V**	1
J	2	**W**	2
K	2	**X**	1
L	2	**Y**	1
M	2	**Z**	2

Tabela 1 - Número de amostras para cada letra do alfabeto aplicadas ao $N.

Foi criada uma janela adicional destinada à seleção do algoritmo a usar: $N, KDA e SVM (Figura 67). Os testes foram realizados de forma a determinar qual deles apresentava maior eficiência no reconhecimento de letras.

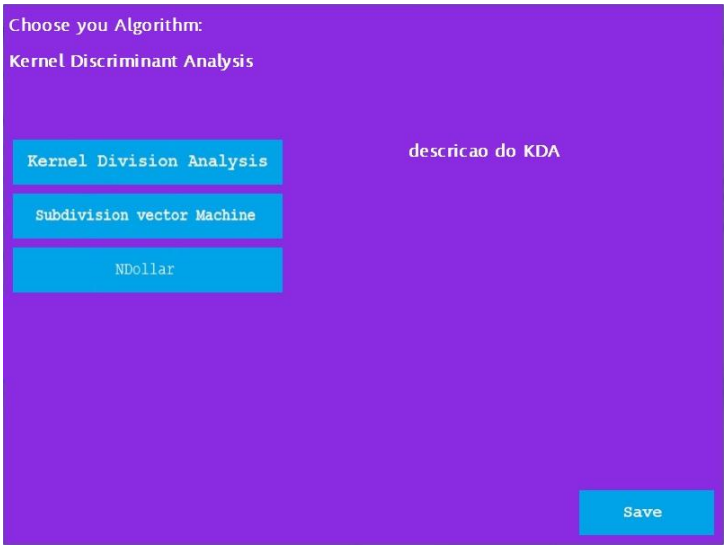


Figura 67 - Ecrã de seleção do algoritmo a usar no reconhecimento.

A função `RecognizeCharacter()` executa o algoritmo selecionado sobre o padrão a reconhecer, resultando deste processo três resultados distintos. Quando não existe correspondência entre o padrão desenhado e as amostras existentes, é dada a indicação de valor desconhecido `"NO MATCH!"`. No caso do valor obtido pelo reconhecedor coincidir com a letra pedida pela aplicação, é dado o resultado `"CORRECT"`. Caso contrário, o resultado é `"INCORRECT"`. Os resultados são constantemente guardados num ficheiro de texto, de forma a permitir, no final, obter a informação dos resultados obtidos, auxiliando assim o processo estatístico.

6.3.8 Gravação do desenho de letras

A aplicação permite que o conjunto de pontos que constitui o desenho realizado, independentemente de ser identificado ou não pelo reconhecedor, seja guardado, associando-lhe pelo utilizador a letra adequada. O conjunto de pontos é então gravado num ficheiro XML, juntamente com os ficheiros das amostras inicialmente usadas no processo de carregamento. Assim, uma identificação inicialmente errada por ausência de uma amostra adequada passa a ser identificada corretamente através destes novos complementos.

6.3.9 Apresentação dos resultados das identificações

Acedendo ao painel de desenho é criado, de modo sequencial, um novo ficheiro de texto onde serão guardados os resultados das identificações dos desenhos produzidos ao longo das cinco tentativas de que o utilizador dispõe. Este ficheiro, contendo os resultados da interação, é iniciado com a *tag* 'START' e termina com 'END', no fim da lista de resultados.

6.3.10 Encerrar aplicação

Para que o utilizador disponha de maior liberdade no modo como navega e interage, evitando que apenas possa encerrar a aplicação no final das cinco tentativas, foi estabelecido um movimento com as duas mãos para a frente, fazendo com que, quando as palmas das mãos se aproximam, a aplicação encerre automaticamente.

6.4 Conclusão

O desenvolvimento do protótipo, permitiu considerar vários aspetos fundamentais neste tipo de aplicações NUI, tais como a definição de limitações para cada tipo de gesto de forma a garantir um nível de precisão considerável ao longo da interação, assim como garantir o correto funcionamento independentemente da estatura do utilizador, mantendo a consistência dos gestos e das ações. Outro aspeto considerado foi a correta transposição dos traços para a tela do monitor usando texturas de pontos em vez de uma matriz de dupla entrada de pontos, reduzindo consideravelmente o custo computacional desta operação.

7 Avaliação

Nesta secção são descritos testes realizados após a conclusão do protótipo desenvolvido, com vista à sua avaliação. São analisados aspetos como a eficiência do reconhecedor de padrões aplicado ao reconhecimento de caracteres e o grau de usabilidade do protótipo. Variou-se o número de amostras aplicadas aos reconhecedores por forma a verificar o grau de eficiência de cada um destes. Ao algoritmo de reconhecimento de padrões $N, foram-lhe aplicados três valores diferentes para o incremento de reamostragem e para o ângulo inicial, por forma a determinar quais as implicações da variação destes parâmetros na taxa de reconhecimento. Por último, procedeu-se a uma averiguação do grau de adaptabilidade e aprendizagem dos dois grupos avaliados.

7.1 Avaliação dos algoritmos de reconhecimento

O algoritmo SVM, quando comparado com o KDA, necessita de um pequeno número de amostras, também designadas de vetores de suporte, ao longo do processo de aprendizagem, diminuindo o tempo de aprendizagem e reduzindo também o nível de complexidade deste processo. Esta necessidade de construção de um modelo de aprendizagem, usado pelo KDA e pelo SVM, faz com que estes algoritmos não sejam aconselháveis para sistemas embebidos.

Relativamente ao número de amostras aplicado a cada um dos algoritmos implementados, foram estipuladas 10 amostras para cada letra do alfabeto, o que perfaz um total de 260 amostras para os algoritmos KDA e SVM. Em contrapartida, o $N usa um conjunto mais restrito, variando consoante o número de linhas que cada letra empregue (secção 6.3.7.2).

Os testes realizados com estes algoritmos basearam-se em 100 desenhos de letras geradas aleatoriamente para cada um dos reconhecedores. Foi possível constatar uma eficácia superior por parte do $N, com uma taxa de reconhecimentos corretos de 92%, contra 63% do KDA e 54% do SVM (Gráfico 1). No caso do $N, a definição de amostras com diferentes ordens de desenho das 26 letras do alfabeto revelou-se essencial para a precisão do reconhecimento.

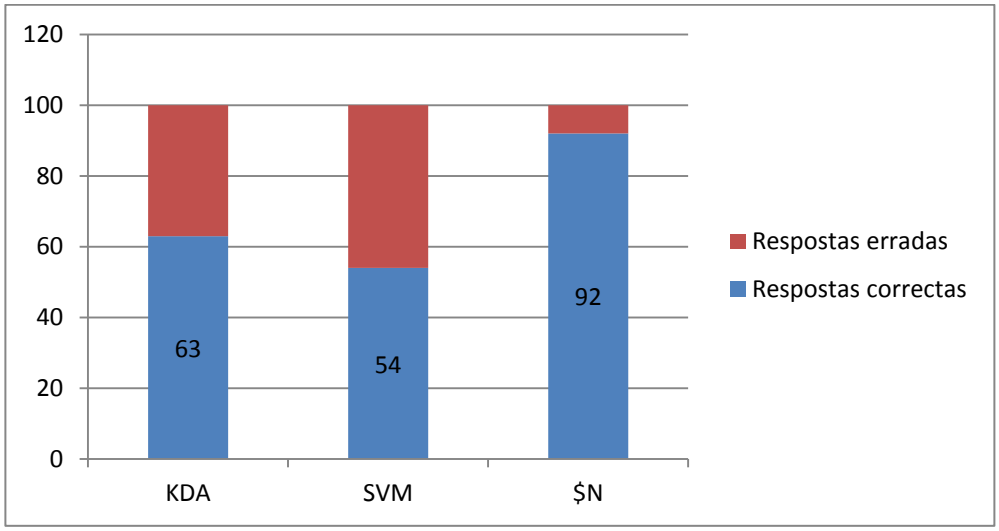


Gráfico 1 - Gráfico comparativo entre os três diferentes algoritmos.

Depois dos testes comparativos entre os diferentes algoritmos, foi por fim selecionado o que demonstrou melhor aptidão: o $N.

Os testes realizados com os diferentes algoritmos basearam a sua análise num conjunto de amostras representativas das 26 letras do alfabeto, apresentado no Gráfico 2. Tal como referido anteriormente, os algoritmos KDA e SVM usaram um conjunto de 260 amostras; já o $N usa um número bastante inferior, recorrendo para o efeito a apenas 41 amostras.

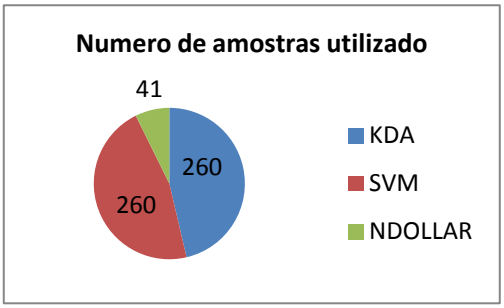


Gráfico 2 - Número de amostras usadas por cada algoritmo implementado.

Foram também realizados testes para averiguar o nível de eficácia de cada um dos algoritmos, recorrendo unicamente a metade dos conjuntos de amostras inicialmente definidos. Nesta verificação, foi possível constatar que a superioridade do $N se mantém face ao KDA e ao SVM, obtendo um total de 73% de respostas corretas. Já os restantes algoritmos ultrapassaram ligeiramente a margem dos 50%, revelando ainda uma falta de eficácia na identificação de letras (Gráfico 3).

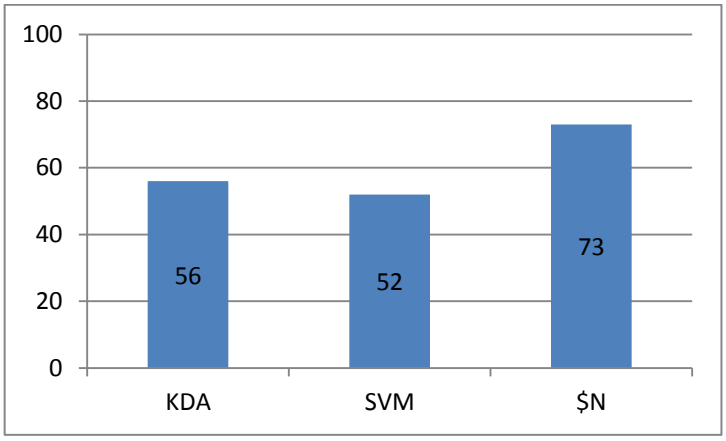


Gráfico 3 - Experiência aplicando metade das amostras definidas.

O algoritmo $N foi sujeito a testes que permitiram ditar quais os melhores parâmetros a configurar, de modo a garantir uma taxa de reconhecimento razoável. As modificações passaram pela alteração do número de pontos no processo de reamostragem em cada conjunto de pontos desenhados. Alterando o valor da reamostragem, o incremento muda, fazendo com que os pontos sejam representados com diferentes espaçamentos entre si, resultando num aumento ou diminuição da definição dos pontos que constituem um traço. No Gráfico 4 encontram-se representados os três diferentes valores aplicados, 16, 64 e 96, num conjunto de 100 tentativas. Verifica-se que a reamostragem usando o incremento 64 é a que oferece maior rigor, tendo sido escolhida para uso pelo reconhecedor da aplicação.

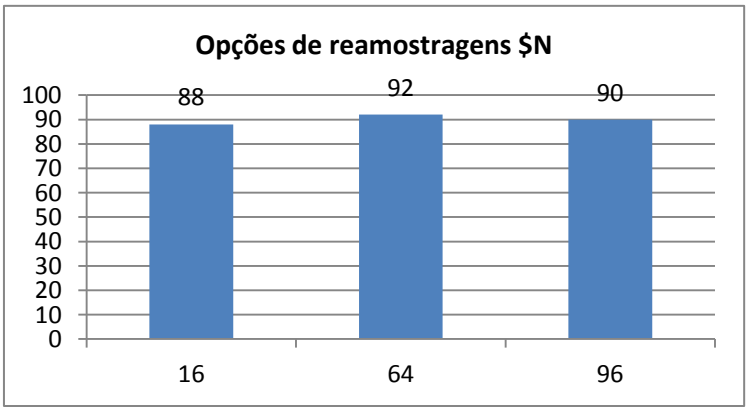


Gráfico 4 - Testes de eficiência do algoritmo $N, usando diferentes valores de reamostragem.

Outro parâmetro considerado foi o ângulo inicial definido para a inicialização do conjunto de pontos desenhados aquando da utilização do método de alinhamento GSS. O ângulo de inicialização é considerado um possível fator de futuros erros na análise. No Gráfico 5, encontram-se representados os valores 30, 45 e 60, num conjunto de 100 tentativas, à semelhança do teste anteriormente descrito do incremento da reamostragem. Pôde-se verificar que o ângulo de 30° é o que oferece maior rigor, tendo sido também escolhido para o reconhecedor na aplicação.

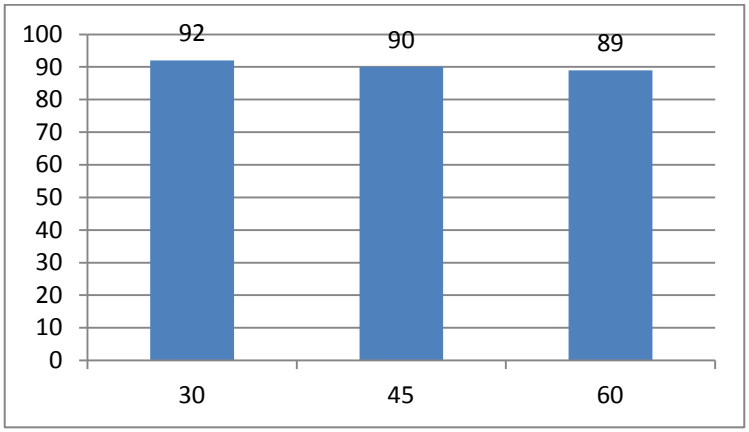


Gráfico 5 - Testes de eficiência do algoritmo $N, aplicando diferentes valores para os ângulos iniciais.

7.2 Avaliação da usabilidade do protótipo

Segundo a avaliação heurística descrita por Jakob Nielsen [Nielsen, 1993], o número de problemas detetados no *design* de uma interface aumenta com o aumento do número de avaliadores. Contudo, indica como valor mínimo aconselhável um número entre três e cinco avaliadores. O exemplo dado por Robert A. Virzi em [Virzi R., 1992] representa um conjunto de anomalias avaliadas por um conjunto de 12 avaliadores inexperientes por forma determinar a taxa que problemas de usabilidade encontrados. Na Figura 68, é possível constatar que é encontrado um maior número de problemas em função do número de avaliadores.

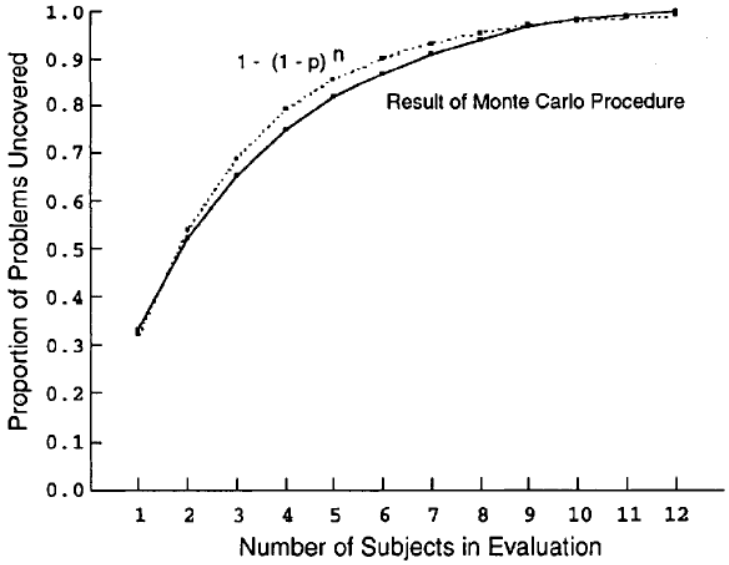


Figura 68 - Função representativa da determinação da usabilidade de um serviço de correio de voz, com base no número de avaliadores.

De modo a verificar a viabilidade da aplicação desenvolvida no âmbito deste projeto, procedeu-se a uma avaliação informal com um grupo de 20 participantes com idades compreendidas entre os 6 e os 25 anos, dos quais 7 elementos eram adultos como termo de comparação, e os restantes crianças do ensino básico, tendo cada uma delas efetuado um conjunto de 10 tentativas.

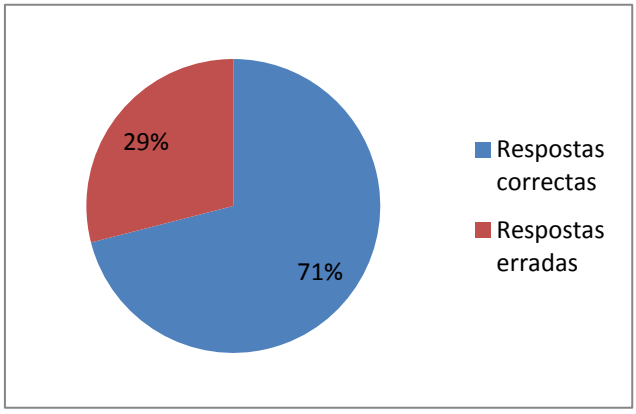


Gráfico 6 - Testes realizados numa classe de alunos do ensino básico.

Destes dois grupos investigados, o grupo mais jovem obteve uma percentagem de respostas corretas de 71.6% (Gráfico 6). Já o grupo de idade mais avançada conseguiu uma percentagem de 93.2% respostas corretas (Gráfico 7).

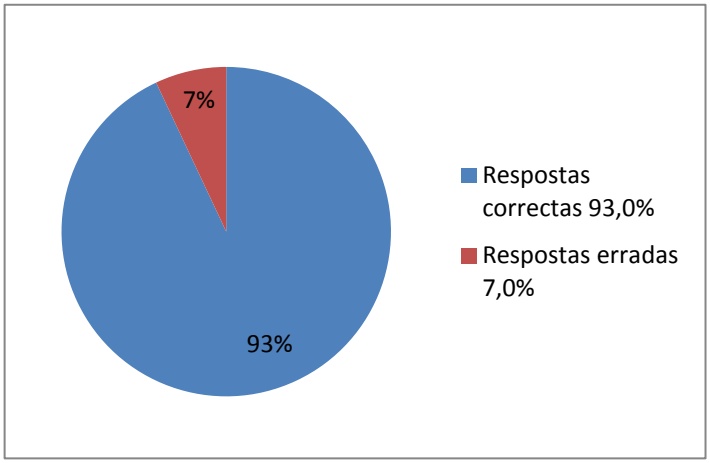


Gráfico 7 - Testes realizados num universo de 7 adultos com idades compreendidas entre os 18 e os 25 anos.

Ao longo dos testes foi possível verificar, por parte dos utilizadores, um certo embaraço nas duas tentativas iniciais, vindo progressivamente a adaptarem-se à interface ao longo das várias experiências realizadas. Considera-se positiva a adaptabilidade do utilizador à aplicação. No entanto, revelou-se essencial sujeitar o utilizador a um processo de aprendizagem. Ao longo das 10 tentativas, o utilizador vai-se familiarizando com os diferentes planos, adquirindo a prática necessária ao desenho das letras. Porém, após apenas três interações, o utilizador

ainda demonstra algumas dificuldades relacionadas com a sensibilidade na transposição dos gestos, bem como dificuldades em situações em que se pretende remover a mão do desenho sem deixar rastos indesejados no ecrã. Estas situações carecem de um treino mais profundo para serem gradualmente corrigidas, melhorando consideravelmente a interação com o utilizador.

7.3 Conclusão

Após o conjunto de testes realizados, o $N foi considerado o algoritmo mais indicado para integrar o protótipo, revelando a sua eficácia mesmo quando é adotado um menor número de amostras. No $N, a definição de amostras com diferentes ordens de desenho das 26 letras do alfabeto revelou-se essencial para a precisão no reconhecimento, obtendo uma taxa de sucesso de 92%, contra 63% do KDA e 54% do SVM. Relativamente à adaptabilidade ao protótipo por parte do público avaliado, considera-se positiva a adaptabilidade por parte destes à aplicação, familiarizando-se progressivamente com os diferentes planos e adquirindo a prática necessária ao desenho das letras ao longo das várias interações.

8 Conclusão e perspetivas de trabalho futuro

O trabalho desenvolvido foca-se no apoio ao processo de ensino e aprendizagem em crianças do ensino básico. Deste modo pensou-se em conceber uma solução de interação simples, com base unicamente nos gestos produzidos pelas crianças ao longo da interação, para desenhar letras e realizando o reconhecimento dos símbolos desenhados. O estudo deste sistema de reconhecimento de gestos no ar, permitiu conceber um protótipo no qual os desenhos produzidos são posteriormente analisados pela aplicação, com uma taxa de reconhecimento bastante positiva, nas identificações efetuadas pelo reconhecedor. Até ao momento, com base nos testes realizados em dois grupos de pessoas de diferentes faixas etárias, foi possível constatar as valias desta aplicação na área da educação e desenvolvimento motor das crianças, assim como diferentes níveis da perceção espacial por parte dos utilizadores das diferentes faixas etárias. Os resultados obtidos indicam que o público-alvo mais jovem demonstra uma maior dificuldade ao longo do processo de familiarização com os planos, em comparação com os utilizadores de idade mais avançada. Este público mais jovem demonstra, em contrapartida, uma maior capacidade de adaptação, melhorando gradualmente as suas capacidades de coordenação.

Um aspeto essencial a considerar para os utilizadores de qualquer faixa etária, prende-se com a necessária familiarização aos planos de interação, por forma a interagir eficientemente com a aplicação.

O protótipo aqui desenvolvido foi divulgado no Vigésimo Encontro Português de Computação Gráfica, despertando a atenção dos participantes pela aplicação do dispositivo Kinect na área do ensino de crianças. O tema foi considerado de grande interesse e relevância para a conferência. O artigo pode ser consultado no apêndice desta dissertação [Leitão et al., 2012].

8.1 Perspetivas de trabalho futuro

Como perspetivas de trabalho futuro, consideraram-se os seguintes aspetos:

8.1.1 Componente gráfica

Na interface da aplicação podem-se realizar diversos melhoramentos, dos quais se destaca a utilização de um modelo dinâmico com um esqueleto interno, constituído por um conjunto de junções que estabeleçam correspondência com as junções captadas pelo Kinect/OpenNI. Desta forma, seria possível animar um modelo inicialmente estático (um *avatar*) com os gestos realizados pelo utilizador.

8.1.2 Componente operacional

Propõe-se adicionar ao protótipo um novo conjunto de amostras, por forma a permitir também o reconhecimento de algarismos e de figuras geométricas.

Seria útil o desenvolvimento de um módulo estatístico para o registo e a análise da evolução do aluno, onde seria apresentado um gráfico com os resultados dos vários testes realizados e respetiva avaliação ao longo de um período a definir pelo administrador do módulo.

Um aspeto fundamental prende-se com o estudo mais aprofundado dos segmentos de interação implementados, por forma a determinar qual a divisão mais indicada em função do comprimento do braço do utilizador.

Referências

[Adafruit, 2011] Adafruit, http://adafruit.com/ [último acesso: Nov 2011

[Adrian and Andy, 2009] Adrian F., Andy S. Features and Future of Open Sound Control version 1.1 for NIME, Department of Music/University of California, 2009.

[Andrew, 2011] Andrew Ng. Support Vector Machines, Machine Learning Autumn 2011 Part V., Stanford University, 2011.

[Anthony and Wobbrock, 2007a] Anthony L., Wobbrock J. $N Multistroke Recognizer in JavaScript. University of Maryland - Baltimore and University of Washington ,USA,2007.

[Anthony and Wobbrock, 2007b] Anthony L., Wobbrock J. A Lightweight Multistroke Recognizer for User Interface Prototypes. University of Maryland - Baltimore and University of Washington ,USA,2007.

[Bespoke Software, 2011] Bespoke Software, http://www.bespokesoftware.org/wordpress/ [último acesso: Nov 2011]

[Berwick and Idiot, 2003] Berwick R., Idiot V., An Idiot's guide to Support vector machines (SVMs), 2003.

[Burrus N., 2011] Burrus N.2012, Kinect Calibration, http://nicolas.burrus.name/index.php/Research/KinectCalibration#tocLink5 [último acesso: Nov 2011]

[Borglabs , 2012] Borglabs, http://borglabs.com/blog/rgb-d-depth-cameras-for-100-in-sight [último acesso: Abr 2012]

[Castaneda and Navab, 2011] Castaneda V., Navab N. Time-of-Flight and Kinect Imaging. Kinect Programming for Computer Vision Summer Term, Computer Aided Medical Procedures, 2011.

[CNMET, 2011] Center for New Music & Audio Technologies, http://cnmat.berkeley.edu/about [último acesso: Nov 2011]

[Code Laboratories, 2011] Code Laboratories, http://codelaboratories.com/kb/nui [último acesso: Nov 2011]

[Dakkak and Husain, 2012] Dakkak A., Husain A. Recovering Missing Depth Information from Microsoft's Kinect, Carnegie Mellon University's, 2012.

[Embedded Vision, 2012] Embedded Vision Alliance, http://www.embedded-vision.com/news/2011/07/27/playstation-move-motion-detection-gets-sony-sanctioned-academic-and-enthusiast-devel [último acesso: Abr 2012]

[Engadget, 2012] Engadget, http://www.engadget.com/2010/12/15/primesenses-openni-provides-the-best-kinect-drivers-yet-from-s/ [último acesso:

	Mar 2012]
[Evansuma, 2012]	Evansuma, 2010. Youtube - World of Warcraft with Microsoft Kinect using FAAST and OpenNI, http://www.youtube.com/watch?v=62wj8eJ0FHw [último acesso: Mar 2012]
[FAAST, 2011]	Flexible Action And Articulated Skeleton, http://projects.ict.usc.edu/mxr/faast/ [último acesso: Nov 2011]
[Fuchs, 2010]	Fuchs, 2010. Ghoshehspft's - 3D Water Bodies, http://ghoshehsoft.wordpress.com/tag/xna-graphics-water-waves/ [último acesso: Mai 2012]
[Gadg, 2012]	Gadg, http://www.gadg.com/2010/11/19/rounding-up-your-room-for-the-new-kinect/ [último acesso: Fev 2012]
[GDIF, 2011]	Gesture Description Interchange Format, http://www.gdif.org/ [último acesso: Nov 2011]
[Geometry of Support Vector Machines, 2012]	Geometry of Suport Vector Machines, http://www.m8j.net/(Math)Geometry%20of%20Support%20Vector%20Machines [último acesso: Abr 2012]
[Halchenko, 2008]	Halchenko Y., 2008 . Iterative Closest Point (ICP) Algorithm. New Jersey Institute of Technology.
[Jacobson, 2012]	Jacobson A., 2009. Alec's Web Log, http://www.alecjacobson.com/weblog/?p=539 [último acesso: Fev 2012]
[Nielsen, 1993]	Jakob Nielsen. Usability Engineering, volume 44. Morgan Kaufmann, 1993 [último acesso: Out 2012]
[Jared, 2011]	Jared, 2011.Kinect-Hacks - Kinect Head Tracking with Rotational Yaw Detection, http://kinect.dashhacks.com/kinect-hacks/2011/09/03/kinect-head-tracking-rotational-yaw-detection [último acesso: Mar 2012]
[Jason and James, 1999]	Jason I. , James A. , SATIN: A Toolkit for Informal Ink-based Applications. Group for User Interface Research, Computer Science Division, University of California, Berkeley, 1999.
[John, 1998]	John P., Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines. 1998.
[Kardi Teknomo's, 2012]	Kardi Teknomo's, http://people.revoledu.com/kardi/tutorial/Regression/KernelRegression/Kernel.htm [último acesso: Mai 2012]
[Kinect Paint, 2012]	Kinect Paint, http://paint.codeplex.com/ [último acesso: Jan 2012]
[Kinect Education, 2012]	Kinect Education, http://www.kinecteducation.com/blog/tag/learning-with-kinect/ [último acesso: Jan 2012]
[Khoshelham, 2011]	Khoshelham K., 2011. Accuracy Analysis Of Kinect Depth Data. ITC Faculty of Geo-information Science and Earth Observation, University of Twente. Netherlands.
[Kjaer, 2011]	Kjaer J., 2011. A Qualitative Analysis of Two Automated Registration Algorithms In a Real World Scenario Using Point Clouds from the Kinect.
[Kolman and Margaliot, 2009]	Kolman E., Margaliot M. Knowledge-Based Neurocomputing: A Fuzzy Logic Approach, Springer, p. 1-12, 2009
[Knies, 2011]	Knies R., 2011. Academics, Enthusiasts to Get Kinect SDK, Microsoft Research.
[Kreylos O. 2012]	Kreylos O., 2012. Kinect Hacking. http://idav.ucdavis.edu/~okreylos/ResDev/Kinect/ [último acesso: Fev 2012]
[Kvamme and Strøm, 2008]	Kvamme T., Strøm J., 2008. Evaluation and Extension of an XNA Game Library used in Software Architecture Projects. Norwegian,

Master of Science in Computer Science.

[kwc , 2012] kwc , 2010. Robot Operating System, http://www.ros.org/news/2010/12/technical-information-on-kinect-calibration.html [último acesso: Jan 2012]

[Leap, 2012] Leap, https://live.leapmotion.com/blog/ [ultimo acesso: Mai 2012]

[Leitão et al., 2012] Leitão P., Pereira P., Castro A. Interface Caligráfica de Escrita no Ar, 20º Encontro Português de Computação Gráfica, 2012.

[Lin et al., 2008] Lin S., Lee Z., Chen S., Tseng T. Parameter determination of support vector machine and feature selection using simulated annealing approach. Elsevier, 2008.

[Lobão et al., 2009] Lobão A., Evangelista B., Farias J., Grootjans R. Beginning XNA 3.0 Game Programming: From Novice to Professional, Apress, 2009.

[Maxwell, 2012] Maxwell, http://www.maxwell.lambda.ele.puc-rio.br/11498/11498_5.PDF [último acesso: Abr 2012]

[Media.zero, 2012] Media.zero, http://media.zero997.com/kinect_shadow.pdf [último acesso: Mar 2012]

[Mehlmann et al., 2011] Mehlmann G., Kistler F., Endrass B., André E., Klimmt C., Wagner J., Smirra R. Analysis of Interactive Storytelling Applications, UK, 2011.

[Microsoft MSDN, 2012a] Microsoft MSDN, http://msdn.microsoft.com/en-us/library/bb197956(v=xnagamestudio.31).aspx [último acesso: Jan 2012]

[Microsoft MSDN, 2012b] Microsoft MSDN, http://msdn.microsoft.com/en-us/library/bb976089(v=xnagamestudio.30).aspx [último acesso: Jan 2012]

[Microsoft MSDN, 2012c] Microsoft MSDN, http://msdn.microsoft.com/en-us/library/bb196414.aspx[último acesso: Mar 2012]

[Microsoft MSDN, 2012d] Microsoft MSDN, http://msdn.microsoft.com/en-us/library/bb203940(v=xnagamestudio.31).aspx [último acesso: Jan 2012]

[Microsoft MSDN, 2012e] Microsoft MSDN, http://msdn.microsoft.com/en-us/library/bb195055(v=xnagamestudio.31).aspx [último acesso: Agos 2012]

[Microsoft TablePC, 2012] Microsoft TablePC, http://www.microsoft.com/en-us/news/features/2000/nov00/11-13tabletpc.aspx [último acesso: Abr 2012]

[Microsoft XNA, 2012] Microsoft XNA, http://www.xnatutorial.com/ [último acesso: Jan 2012]

[Morvan, 2009] Morvan Y., 2009. Acquisition, Compression and Rendering of Depth and Texture for Multi-View Video. Ph.D. Thesis, Eindhoven University of Technology. The Netherlands.

[NIME, 2011] New Interfaces for Musical Expression, http://www.nime.org/ [último acesso: Nov 2011

[O'Neill, 2006] O'Neill M., 2006.Neural Network for Recognition of Handwritten Digits.Code Project, United States.

[OpenKinect, 2012] OpenKinect, http://openkinect.org/wiki/Imaging_Information [último acesso: Fev 2012]

[OpenKinect, 2011] OpenKinect, http://openkinect.org/wiki/Main_Page [último acesso: Out 2011]

[OpenNI, 2011a] OpenNI, http://75.98.78.94/images/stories/pdf/OpenNI_UserGuide_v4.pdf%20%5b1 [último acesso: Out 2011]

[OpenNI, 2011b] OpenNI, http://openni.org/docs2/ProgrammerGuide.html [último acesso: Out 2011]

[OpenNI, 2011c] OpenNI, http://openni.org/docs2/ProgrammerGuide.html [último

	acesso: Nov 2011]
[OpenSoundControl, 2011]	OpenSoundControl, http://opensoundcontrol.org/introduction-osc [último acesso: Nov 2011]
[O'Reilly Answers, 2012]	O'Reilly Answers, http://answers.oreilly.com/topic/2443-introduction-to-openkinect-and-as3kinect/ [último acesso: Mar 2012]
[O'Reilly , 2012]	O'Reilly - Ink: Handwriting Recognition, Chapter 15, http://examples.oreilly.com/0636920014553/ink_handwriting_recognition.pdf[último acesso: Mar 2012]
[Osuna, 2012]	Osuna R., Pattern Analysis. Texas A&M University, USA.
[OSCeleton, 2011]	OSCeleton, https://github.com/Sensebloom/OSCeleton [último acesso: Nov 2011]
[Palasoftware, 2012]	Palasoftware, http://www.palasoftware.com/Home.html [último acesso: Abr 2012]
[PCB, 2012]	Pointcluds, http://pointclouds.org/ [último acesso: Abr 2012]
[Perlroth, 2010]	Perlroth N., 2010. http://www.forbes.com/sites/nicoleperlroth/2010/12/13/for-primesense-microsofts-kinect-is-just-the-beginning/ [último acesso: Jan 2012]
[Pinto, 2012a]	Pinto P., 2012. CES 2012 – Microsoft anuncia Kinect para Windows. http://pplware.sapo.pt/informacao/ces-2012-microsoft-anuncia-lanamento-do-kinect-para-windows/ [último acesso: Mar 2012]
[Pinto, 2012b]	Pinto P., 2012. SDK v1.5 do Kinect para Windows disponível para download. http://pplware.sapo.pt/windows/software/sdk-v1-5-do-kinect-para-windows-disponvel-para-download/ [último acesso: Mai 2012]
[Plamondon et al., 2000]	Plamondon R. IEEE, Sargur N., On-line and Off-line Handwritting Recognition: A Comprehensive Survey, USA, 2000.
[PrimeSense, 2012a]	PrimeSense, http://www.primesense.com/en/making-history [último acesso: Jan 2012]
[PrimeSense, 2012b]	PrimeSense, http://www.primesense.com/en/nite [último acesso: Jan 2012]
[PrimeSense, 2011c]	PrimeSense, http://andrebaltazar.files.wordpress.com/2011/02/nite-controls-1-3-programmers-guide.pdf [último acesso: Out 2011].
[PStamirb, 2012]	PStamirb, 2010. Youtube - OpenNI-compliant real time SceneAnalyzer by PrimeSense, http://www.youtube.com/watch?v=-_j7BzSmlfU [último acesso: Mar 2012]
[Rabello et al., 2008]	Rabello B., Mattos E., Evangelista B., Clua E. Introdução ao XNA, p. 39-82, 2008.
[Virzi R., 1992]	Robert A Virzi. Refining the test phase of usability evaluation: How many subjects is enough? Human Factors, 34(4):457–468, 1992 [último acesso: Out 2012]
[Rogers, 2011]	Rogers R., 2011. Journal. Kinect with Linux. Houston.
[Schwarz, 2011]	Schwarz L., 2011. Lab Course Kinect Programming for Computer Vision. Kinect Lab Course, Computer Aided Medical Procedures.
[SenseBloom, 2011]	SenseBloom, 2010. Youtube - XBox Kinect OSCeleton, http://www.youtube.com/watch?v=led3Vxg7zKo [último acesso: Nov 2011]
[SideKick, 2011]	SideKick, http://www.sidekick.co.il/ [último acesso: Out 2011]
[Souza, 2012a]	Souza C., 2009. Principal Component Analysis in C#. http://crsouza.blogspot.pt/2009/09/principal-component-analysis-in-c.html [último acesso: Abr 2012]
[Souza, 2012b]	Souza C., 2010. Linear Discriminant Analysis in C#. http://crsouza.blogspot.pt/2010/01/linear-discriminant-analysis-in-c.html [último acesso: Abr 2012]

[Souza, 2010a] Souza C., 2010. Handwriting Recognition Revisited: Kernel Support Vector Machines. The Code Project. Brazil.

[Souza, 2012c] Souza C., 2010. Kernel Principal Component Analysis in C#, http://crsouza.blogspot.pt/2010/01/kernel-principal-component-analysis-in.html [último acesso: Abr 2012]

[Souza, 2012d] Souza C., 2010. Kernel Support Vector Machines for Classification and Regression in C#, http://crsouza.blogspot.pt/2010/04/kernel-support-vector-machines-for.html [último acesso: Abr 2012]

[Souza, 2010e] Souza C., 2010. Handwriting Recognition Revisited: Kernel Support Vector Machines. The Code Project. Brazil.

[Souza, 2012f] Souza C., 2010. Kernel Discriminant Analysis in C#, http://crsouza.blogspot.pt/2010/01/kernel-discriminant-analysis-in-c.html [último acesso: Abr 2012]

[Skeletonmaker, 2012] Skeletonmaker, http://code.google.com/p/skeletonmaker/ [último acesso: Abr 2012]

[Spong, 2012] Spong, http://spong.com/asset/322811/11/21059/Microsoft-Natal-Based-On-Low-Cost-Plug-and-Play-Usb-Powered-Device [último acesso: Abr 2012]

[Sylverberg et al., 2007] Sylverberg T., Kristensson P., Leifler O., Berglund E., Drawing on Paper Maps: Reliable On-line Symbol Recognition of Handwritten Symbols Using a Digital Pen and a Mobile Phone, 2007.

[Support Vector Machine Regression, 2012] Support Vector Machine Regression, http://kernelsvm.tripod.com/ [último acesso: Abr 2012]

[Tanaka et al., 2012] Tanaka K., Parker J., Baradoy G., Sheehan D., Holash J., Katz L. A Comparison of Exergaming Interfaces for Use in Rehabilitation Programs and Research, Vol 6, 2012.

[Tscherrig, 2011] Tscherrig J., 2011. Activity Recognition using Kinect.

[TUIO, 2011] Tangible User Interface Objects, http://www.tuio.org/ [último acesso: Nov 2011]

[udiprod, 2012] Udiprod , 2007. Youtube - SVM with polynomial kernel visualization, http://www.youtube.com/watch?v=3liCbRZPrZA&feature=player_embedded [último acesso: Abr 2012]

[Villaroman et al.,2011] Villaroman N., Rowe D., Swan B., Teaching natural user interaction using OpenNI and the Microsoft Kinect sensor. Proceeding SIGITE '11 Proceedings of the 2011 conference on Information technology education. p. 227-232, USA, 2011.

[Wigdor and Wixon, 2011] Wigdor D., Wixon D. Brave NUI World: Designing Natural User Interfaces for Touch and Gesture, Morgan Kaufmann, 2011.

[Wiimotelib, 2012] Managed Library for Nintendo's Wiimote, http://wiimotelib.codeplex.com/ [último acesso: Mar 2012]

[Willow Garage, 2011] Willow Garage, http://www.willowgarage.com/blog?page=8 [último acesso: Out 2011]

[Wobbrock et al., 2007b] Wobbrock J., Wilson A., Li Y. $1 Unistroke Recognizer in JavaScript. University of Washington and Microsoft Research, USA, 2007.

[Wobbrock et al., 2007a] Wobbrock J., Wilson A., Li Y. Gestures without Libraries, Toolkits or Training:A $1 Recognizer for User Interface Prototypes. University of Washington and Microsoft Research, USA, 2007.

[WonwooLee06 , 2012] WonwooLee06 , 2011.Youtube - Kinect Color - Depth Camera Calibration, http://www.youtube.com/watch?feature=player_embedded&v=wMAJ95oQpnE [último acesso: Mar 2012]

[XEN, 2012] Xen: Graphics API for XNA, http://xen.codeplex.com/ [último acesso: Jan 2012]

[Yongmin et al., 2001] Yongmin L., Shaogang G., Liddell H.,Kernel Discriminant Analysis. University of London, Department of Computer Science, UK, 2001

Apêndices

Interface Caligráfica de Escrita no Ar

Pedro Leitão
Dep. Eng. Informática,ISEP-Instituto Politécnico do Porto
R. Dr. António Bernardino de Almeida, 431 Porto
pedromi-guel_rs70@hotmail.com

António Castro
GILT/ISEP-Instituto Politécnico do Porto
R. Dr. António Bernardino de Almeida, 431 Porto
avc@isep.ipp.pt

João J. Pereira
GECAD/ISEP-Instituto Politécnico do Porto
R. Dr. António Bernardino de Almeida, 431 Porto
jjp@isep.ipp.pt

Resumo

Na actualidade, está a emergir um novo paradigma de interacção, designado por Natural User Interface (NUI), para reconhecimento de gestos produzidos com o corpo do utilizador. O dispositivo de interacção Microsoft Kinect, inicialmente concebido para controlo de videojogos, demonstra ser uma aposta viável para explorar outras áreas, como a do apoio ao processo de ensino e de aprendizagem para crianças do ensino básico. O protótipo desenvolvido visa definir um modo de interacção baseado no desenho de letras no ar, e realizar a interpretação dos símbolos desenhados, usando os reconhecedores de padrões KDA, SVM e $N.

Abstract

A new interaction paradigm, named Natural User Interface (NUI), for the recognition of gestures produced with the user's body is emerging. The interaction device Microsoft Kinect, which was originally designed to control video games, proves to be a viable wager to explore other areas such as support to the teaching and learning process for children of elementary school. The developed prototype aims to define a mode of interaction for drawing letters in the air, and perform the interpretation of the symbols by means of the pattern recognizers KDA, SVM and $ N.

Keywords

Classificadores estatísticos, NUI, Kinect, Nuvens de pontos, Calibração, OpenNI, NITE, XNA, OSC, $N.

1. INTRODUÇÃO

A interacção com equipamentos electrónicos usando teclado, rato ou outro controlador manual, tem vindo a ser bastante vulgarizada. O dispositivo Microsoft Kinect pretende colmatar as limitações humano-máquina existentes, através de uma interacção baseada em gestos realizados com o corpo. Este dispositivo, inicialmente desenvolvido para a utilização em videojogos, tem vindo a despoletar interesse junto do público e a demonstrar diversas potencialidades. A aplicabilidade do Kinect não se limita unicamente aos videojogos, existindo um vasto leque de áreas nas quais pode ser utilizado. O âmbito deste projecto visa a aplicação do Kinect, juntamente com o reconhecimento de padrões, à área do ensino, integrando os algoritmos KDA, SVM e $N de forma a interpretar e identificar as letras correspondentes aos símbolos desenhados pelo utilizador, auxiliando assim o desenvolvimento motor e apoiando o processo de aprendizagem.

Nos tempos que correm, com a globalização e o crescimento tecnológico acelerado, as pessoas têm vindo a aderir cada vez mais a equipamentos digitais, em detrimento da escrita convencional (papel e lápis). Contudo, e apesar de a sociedade se encontrar cada vez mais informatizada e dependente da tecnologia, a escrita convencional continua a ter um papel indispensável no quotidiano das pessoas, sobrepondo-se, em determinadas situações, à escrita digital. A escrita de textos, equações ou o desenho de esquemas nas salas de aulas constituem um bom exemplo da sua indispensabilidade. Estas tarefas tornam-se mais funcionais quando aliadas a uma interface tradicional.

Ao longo do tempo têm surgido diversas aplicações com este tipo de interfaces, refere Tomas Sylverberg [Sylverberg et al., 07]. Este autor propõe, como forma de apoio a militares, o uso de dispositivos móveis que lhes permitam desenhar símbolos NATO[1], essenciais na orientação dos soldados no terreno. O uso de dispositivos permitiria a partilha de informações do terreno, reportando a informação desenhada aos restantes elementos dispersos no campo.

[1] Conjunto de símbolos militares padronizados, para marcações em mapas pela NATO.

Nas secções 2 e 3 serão apresentados respectivamente alguns dos principais dispositivos NUI existentes no mercado e as correspondentes bibliotecas de desenvolvimento. A secção 4 aborda os algoritmos de reconhecimento de padrões analisados no âmbito deste projecto. O trabalho realizado e a respectiva avaliação são descritos nas secções 5 e 6, respectivamente. Por último, na secção 7, procede-se à conclusão do artigo e traçam-se as perspectivas de trabalho futuro.

1. HARDWARE

Com a mudança do paradigma de interacção, várias marcas de consolas de videojogos estão a lançar para o mercado novos produtos inovadores na área da interacção natural.

1.1 Wii

Nos anos 80, a Nintendo tentou, pela primeira vez, mudar o modo de interacção dos seus produtos com a criação da *Nintendo PowerPad*. Contudo, apenas anos mais tarde, em 2006, com o lançamento da *Wii*, o público aderiu consideravelmente a este modo de jogo. A *Wii* vem equipada com um sensor e um dispositivo semelhante a um comando de televisão, designado por *Wii Remote*, que utiliza um microcontrolador[1] ADXL330. Este controla três acelerómetros em simultâneo, um para cada eixo, de forma a obter a velocidade e a posição da mão do jogador [Tanaka et al., 12].

1.2 PlayStation Move

A multinacional japonesa *Sony* lançou para o mercado, no ano de 2004, o dispositivo de captura visual baseado em gestos denominado *Eye Toy*. Este conseguiu alcançar alguma notoriedade, despertando o interesse do público por este novo modo de interacção, apesar da baixa capacidade de captura de informação bidimensional de que dispunha. A *PlayStation Move* inclui o *Move Eye*, que consiste numa câmara RGB com uma resolução máxima de 640 por 480 píxeis, a funcionar a uma frequência de 60 a 120 *frames* por segundo (FPS). Contém também um conjunto de microfones e um dispositivo, designado por *Motion Controller*, idêntico a um bastão com uma esfera luminosa. O reconhecimento dos movimentos realiza-se com base na obtenção das coordenadas tridimensionais da esfera luminosa pela câmara RGB [Tanaka et al., 12].

1.3 Microsoft Kinect

O Kinect surgiu de uma conferência realizada em Junho de 2009 (Figura 1), sendo que o seu lançamento para o mercado aconteceu apenas em Novembro de 2010 [Tscherrig, 11]. O projecto que esteve na origem do Microsoft Kinect designou-se por 'Projecto Natal' e teve como base a tecnologia PrimeSense. Adoptava um sistema de luz padronizada e o microcontrolador PS1080 SoC, concebendo assim um dispositivo de controlo de videojogos para a consola Xbox 360 da Microsoft. Na sua constituição podemos encontrar um conjunto de microfones para aplicações que envolvam o reconhecimento de comandos por voz, uma câmara RGB com uma frequência de amostragem de 30 FPS e uma resolução de 640x480 píxeis. Além desta, inclui uma câmara de infravermelhos (IR) com uma resolução de 640x480 píxeis e frequência de amostragem de 30 FPS, usada para medir a profundidade dos elementos inseridos no cenário através da luz estruturada projectada sob eles e posteriormente reflectida para a câmara IR.

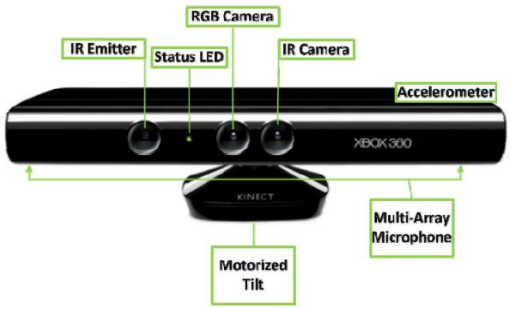


Figura 1 - Constituição do Microsoft Kinect [Mehlmann et al., 11]

2. SOFTWARE

A comunicação e acesso às funções do dispositivo Kinect processam-se através de um *driver*[2] de comunicação entre o dispositivo e o sistema operativo. Os vários *drivers* de comunicação existentes, garantem o acesso a diferentes funcionalidades e suporte para diferentes sistemas operativos.

2.1 Microsoft Kinect SDK

A 21 de Maio de 2012 foi disponibilizada a versão comercial 1.5 da SDK Kinect. Esta versão contém um conjunto de ferramentas pensado para programadores e empresas, permitindo-lhes desenvolverem aplicações para este dispositivo. Esta versão comercializável veio permitir tanto a empresas como ao público em geral usufruírem de um conjunto de ferramentas para Windows direccionadas para o dispositivo Kinect [Pinto, 12b].

2.2 OpenNI

A plataforma OpenNI é uma plataforma para dispositivos NUI, que surgiu em Dezembro de 2010, fruto da associação de empresas sem fins lucrativos como a Willow Garage, desenvolvedora de hardware e software livre para aplicações pessoais robóticas [Willow Garage, 11], a Side-Kick, líder no desenvolvimento de jogos e publicação especializada de videojogos baseados em NUI [SideKick, 2011], e a fabricante de equipamentos informáticos ASUS, que comercializa o dispositivo NUI WaviXtion, baseado na tecnologia PrimeSense [Rogers, 11]. Através de um esforço conjunto estabeleceram uma plataforma padrão para equipamentos NUI, assegurando a compatibilidade, interoperabilidade e a comunicação entre estes equipamentos [OpenNI, 11a].

A sua infra-estrutura é segmentada por camadas (Figura 2), nas quais disponibiliza a interface para módulos de

[1] Consiste num computador num *chip*, contendo um processador, memória e periféricos de entrada/saída.

[2] Pequeno programa que faz a comunicação entre o sistema operativo e o hardware.

dispositivos físicos e módulos para *middleware* de comunicação, associando-os à plataforma e acabando assim com problemas de incompatibilidades entre *middleware* e sensores ou vice-versa. Com isto, passa a ser possível desenvolver aplicações e portá-las para funcionar com outro *middleware* ou dispositivo sem que sejam necessárias modificações suplementares.

As camadas que constituem a infra-estrutura OpenNI são as que a seguir se discriminam [OpenNI, 11b]:

1. A camada aplicação funciona com base nos dados fornecidos da camada inferior e permite interagir com aplicações NUI alto nível;
2. Camada que representa as interfaces OpenNI, e os vários *middlewares* que podem ser associados à plataforma. Tem como objectivo processar os dados adquiridos pelos dispositivos e traduzi-los para serem interpretados pela camada mais acima;
3. Camada responsável por fazer a associação à plataforma dos diferentes dispositivos de captura de informação visual e áudio.

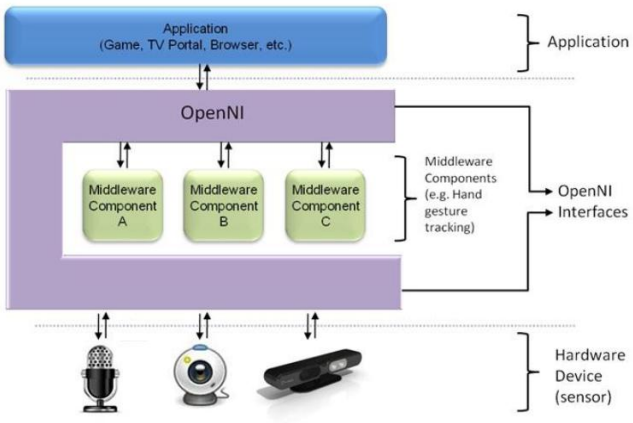


Figura 1 - Estrutura de camadas OpenNI [OpenNI, 11b]

1. ALGORITMOS DE RECONHECIMENTO CALIGRÁFICO

O reconhecimento de caracteres insere-se na área do reconhecimento de padrões, analisando características de uma imagem e decompondo-a de modo a extrair-lhe todas as particularidades que a caracterizam relativamente a outras amostras [Maxwell, 12]. Os reconhecedores dividem-se em dois grupos: estatísticos e coneccionistas. Os classificadores estatísticos, nos quais se incluem os algoritmos KDA, SVM, $1 e $N, recorrem a um conjunto de amostras para aprendizagem antes do processo de classificação. Os classificadores coneccionistas englobam algoritmos de redes neuronais e tentam simular as intercomunicações existentes entre os neurónios existentes num sistema neuronal biológico [Kolman and Margaliot, 09].

1.1 Kernel Discriminant Analysis

O KDA é uma extensão do algoritmo linear estatístico Linear Discriminant Analisys (LDA), usado no reconhecimento de padrões e aprendizagem supervisionada [Yongmin et al., 01]. O KDA transpõe um problema para um espaço dimensional, representando as amostras através de vectores no espaço de características. Posteriormente, usando uma técnica Kernel, maximiza e determina não-linearmente a combinação de características representadas pelos vectores organizados em classes.

1.2 Support Vector Machine

O Support Vector Machine (SVM) é um algoritmo linear estatístico, descoberto por Vladimir Vapnik em 1963, aplicado no reconhecimento de padrões e aprendizagem supervisionada [Souza, 10b]. Este algoritmo procede a uma classificação linear binária dos dados de entrada, separando-os em duas categorias. O modelo de dados definido pelo SVM mapeia o conjunto de características através de vectores no espaço de características, separando-os em duas categorias, permitindo assim distinguir binariamente cada característica com base na categoria correspondente.

1.3 $1

O $1 é um método estatístico não-supervisionado conciso e simples, com uma taxa assinalável de reconhecimento de gestos [Anthony and Wobbrock, 2007a].
Funciona com base em geometria simples e trigonometria, que o torna simples e funcional, fazendo com que possa ser implementado em sistemas de prototipagem rápida. Neste são definidos dois conjuntos: o conjunto de pontos desenhados no momento para reconhecimento, designado de conjunto candidato *C*; e as várias amostras de pontos existentes T_i, que servem como modelos de comparação. O $1 caracteriza-se pela sua total invariância[1] no que respeita à orientação dos gestos, fazendo com que, por exemplo, os sinais '>' e '<', sejam considerados como sendo o mesmo símbolo no processo de reconhecimento, não obstante as suas diferentes orientações.

A comparação entre os dois conjuntos, *C* e T_i, $\boldsymbol{T_i}$aplica o cálculo da distância euclidiana para determinar a melhor aproximação dos conjuntos de pontos existentes em T_i e que oferecem melhor alinhamento com o conjunto candidato *C* desenhado. Assim, com base nesta aproximação, é obtido o valor da menor distância d_i entre os vários conjuntos testados e quantificados, usando um sistema de pontuações, sendo que o que obtiver maior pontuação será o conjunto seleccionado [Wobbrock et al., 07b].

Antes do processo de procura e alinhamento do conjunto *C*, o caminho de pontos definidos por este é sujeito a quatro operações. A primeira passa por tornar os pontos de *C* equidistantes, reamostrando os pontos segundo um processo iterativo, no qual o distanciamento dos pontos varia consoante esse incremento. No segundo passo, procura rodar *C* através do ponto centróide[2] definido pelo caminho dos pontos e pelo ponto em que se inicia o caminho dos pontos, realizando uma rotação para o ângulo indica

[1] Insensibilidade à rotação.

[2] Ponto no interior de uma forma geométrica que define o seu centro geométrico.

tivo (0º) com base nestes dois pontos. Além de se encontrar o ângulo indicativo do conjunto *C*, são também procurados os ângulos indicativos dos conjuntos T_i aquando do seu carregamento, permitindo assim aperfeiçoar a procura do conjunto que melhor se alinha usando o método de alinhamento GSS[1].

O terceiro passo resume-se a redimensionar não uniformemente o caminho de pontos para um quadrado tido como de referência, com um comprimento dos lados ***size*** igual em todas as amostras de T_i. Por fim, realiza a translação do centróide de *C* ***C***para a origem (*x*, *y*) = (0, 0). Estes passos têm como intuito normalizar todos os gestos desenhados de forma a optimizar *C*, facilitando a correspondência dos pontos e o alinhamento com T_i, usando o GSS, como se pode observar na Figura 3 [Wobbrock et al., 07b].

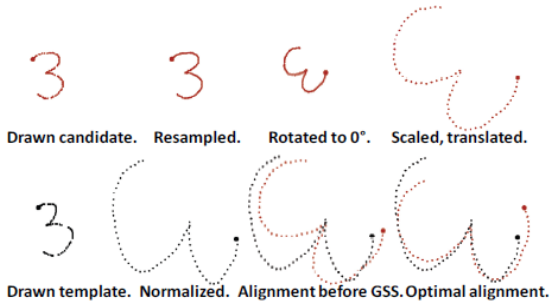


Figura 1 - Descrição dos passos de correspondência e alinhamento de um conjunto de pontos efectuado pelo $1 [Anthony and Wobbrock, 07b]

1.1 $N

O $N baseia o seu funcionamento na mesma estratégia definida pelo seu predecessor $1, realizando um conjunto de operações com base em geometria simples e trigonometria. Contudo, a grande melhoria encontra-se na versatilidade e capacidade de reconhecimento de gestos compostos por múltiplos traços [Anthony and Wobbrock, 07a].

Foram tidas em consideração duas questões, também abordadas por [Anthony and Wobbrock, 07b], que se prendem com a articulação dos gestos realizados e o efeito do número de amostras no rigor do reconhecimento. Relativamente à velocidade de articulação em que os gestos se processam, um aumento da velocidade não implica um aumento da taxa de erros no processo de reconhecimento, visto a distribuição dos pontos variar consoante o incremento do processo de reamostragem. Já o aumento do número de amostras melhora consideravelmente a precisão dos reconhecimentos.

O $N foi sujeito a testes que permitiram determinar quais os melhores parâmetros a configurar, de forma a garantir uma taxa de reconhecimento razoável. As modificações passaram pela alteração do número de pontos no processo de reamostragem em cada conjunto de pontos desenhados. A modificação do valor da reamostragem faz com que cada ponto seja representado com os espaçamentos entre pontos definido. Como resultado, ocorre um aumento ou diminuição da clareza dos pontos que constituem um traço. No gráfico da Figura 4 estão representados os 3 diferentes valores aplicados, 16, 64 e 96, num conjunto de 100 tentativas. Verifica-se que a reamostragem usando o incremento 64 oferece maior rigor.

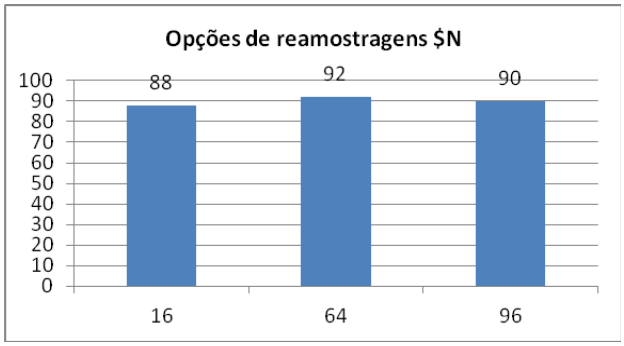


Figura 2 - Testes de eficiência do algoritmo $N, usando diferentes valores de reamostragem

2. DESENVOLVIMENTO DA APLICAÇÃO

O software, desenvolvido em ambiente Windows, sobre a plataforma XNA Game Studio 3.1, visa definir uma interface de desenho de letras no ar que reconheça letras desenhadas por crianças do ensino básico. O dispositivo NUI Microsoft Kinect é usado como base para a concepção da aplicação, seguindo uma ordem de trabalhos que começa por localizar as articulações do corpo do utilizador, definir o plano de escrita e desenho das letras e, por fim, incorporar algoritmos de reconhecimento de padrões, para identificar as letras desenhadas.

Apesar de a versão oficial Microsoft Kinect SDK já se encontrar disponível actualmente, aquando do início deste projecto esta livraria estava apenas disponível na versão experimental, razão que levou a optar pelo uso da plataforma OpenNI. Outros factores influenciaram esta decisão, tais como o funcionamento com diferentes sistemas operativos, ser uma plataforma padronizada para qualquer equipamento, e tratar-se de uma plataforma de uso livre sujeita a constantes estudos, o que contribui para o aumento da sua maturidade e estabilidade.

Foram realizados testes com diferentes algoritmos de reconhecimento de padrões, que basearam a sua análise num conjunto de amostras representativas das 26 letras do alfabeto latino. Ambos os KDA e SVM usaram um conjunto de 260 amostras. Já o $N usou um número bastante inferior, recorrendo para o efeito a apenas 41 amostras. A simplicidade e a flexibilidade na identificação de padrões, aliado ao reduzido número de amostras, fazem do $N a melhor opção no processo de reconhecimento de padrões.

[1] Técnica que consiste em encontrar o limite máximo ou mínimo recorrendo a uma função que analisa a distribuição dos valores dentro desses limites.

1.1 Interface com o utilizador

A aplicação desenvolvida pretende criar uma analogia com o modo de desenho de símbolos na areia da praia. Esta aplicação está dividida em seis grupos: detecção das articulações, configuração do painel de escrita, desenho de letras, reconhecimento de letras, gravação do desenho de letras e apresentação dos resultados das identificações.

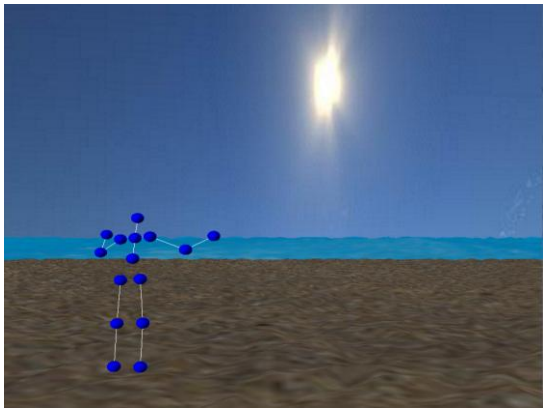


Figura 1 - Ecrã de localização das articulações do utilizador

O primeiro grupo, da detecção das articulações, destina-se à localização e rastreio do utilizador. Para isso, este deverá levantar os dois braços e baloiçá-los alternadamente para a esquerda e para a direita, até que o sistema o detecte e lhe reconheça as articulações do corpo. As articulações obtidas são posteriormente representadas através de esferas ligadas por intermédio de linhas, representando o esqueleto do utilizador e apresentando-o no cenário inicial da praia (Figura 5).

Figura 2 - Ecrã de indicação ao utilizador, destinado à selecção da mão de desenho que pretende usar

Depois de localizar o corpo, e ainda no cenário inicial da praia, segue-se a operação de Configuração do painel de escrita. Aqui, o utilizador deve estabelecer a distância entre si e o plano de escrita, movendo a mão direita para o lado direito, aproximadamente à mesma altura do ombro, e de seguida baixando a mão esquerda abaixo do nível da anca esquerda. Após definida a distância do utilizador ao plano de escrita, aquele deve indicar qual das mãos pretende usar. Assim, quando aparece o ecrã ilustrado na Figura 6, deverá projectar para a frente a mão pretendida.

Figura 3 - Interface do ecrã de desenho

Depois de determinar os parâmetros inerentes à configuração do painel de desenho, é iniciada uma nova fase (Figura 7). Aí são concedidas cinco tentativas, nas quais o utilizador deverá acertar em cada uma das letras pedidas. Estas são geradas aleatoriamente de acordo com as 26 letras do alfabeto. O utilizador poderá movimentar o cursor (uma representação de um galho de uma árvore), movendo a mão de desenho ao longo de um plano situado mais próximo do seu corpo, ou então movimentar o cursor ao mesmo tempo que vai adicionando pontos ao ecrã, tendo a mão seleccionada de atravessar o plano mais afastado. Terminado o desenho estão disponíveis duas opções: a de limpar os pontos adicionados ou a de identificar o símbolo desenhado.

Figura 4 - Ecrã de selecção da letra a atribuir ao símbolo desenhada

Para identificar o conjunto de pontos que constituem o símbolo desenhado, surge um novo cenário para o grupo do reconhecimento de letras, onde é apresentado o resultado da identificação do símbolo, realizada pelo reconhecedor de padrões $N. Aqui, estão unicamente disponíveis as opções para guardar ou não o referido símbolo.

Caso não se pretenda guardar dever-se-á seleccionar a opção 'Não', sendo automaticamente direccionado para o grupo de desenho de letras (o ecrã anterior, representado na Figura 7). Caso contrário, se seleccionar a opção para guardar o desenho, é então que surge um novo cenário (Figura 8), apresentando as 26 letras do alfabeto possíveis de associar ao conjunto de pontos do desenho que se pretende guardar. Para seleccionar a letra pretendida, o utilizador deverá projectar a mão de desenho até ao plano mais afastado, escolher a letra e, por fim, accionar o botão para guardar. Esta adição de um novo conjunto de pontos ao conjunto de amostras existentes enriquece o processo de reconhecimento e generalização de futuros desenhos.

Por fim, após terminarem as cinco tentativas disponíveis para o utilizador responder às letras geradas pela aplicação, são mostrados os resultados da identificação de cada símbolo pelo algoritmo de reconhecimento de padrões. As letras correctamente identificadas são assinaladas com um "visto" verde. Os erros são marcados com uma "cruz" vermelha.

1.1 Detalhes técnicos

De forma a obter as coordenadas das articulações, o utilizador deve começar por indicar a sua presença perante o dispositivo Microsoft Kinect (Figura 9), como referido anteriormente, até que as articulações sejam detectadas.

Figura 1 - Detecção da presença do utilizador, acenando com os braços acima da cintura

A definição do plano de escrita consiste, na realidade, num plano de dupla profundidade, estabelecido com base na medição da distância medida da mão direita ao ombro direito do utilizador (Figura 10).

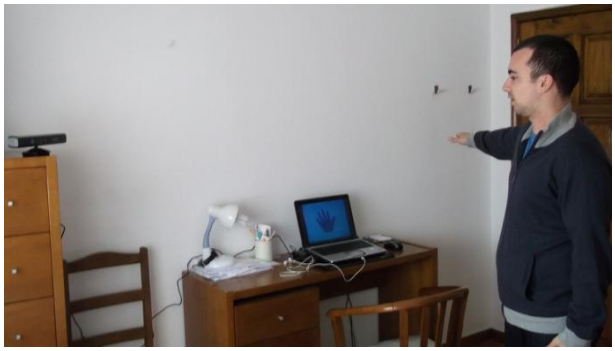


Figura 2 - Postura definida para a determinação do duplo plano de interacção

As áreas de interacção no ar foram pensadas como um conjunto de três segmentos, definidos em função do valor obtido da medição referida anteriormente. Esse valor é segmentado em três (Figura 11), ficando assim o segmento que vai dos 0% aos 25% a tratar-se de um segmento de ócio. Assim, sempre que a mão de escrita se encontrar dentro deste segmento, não será desencadeada qualquer acção.

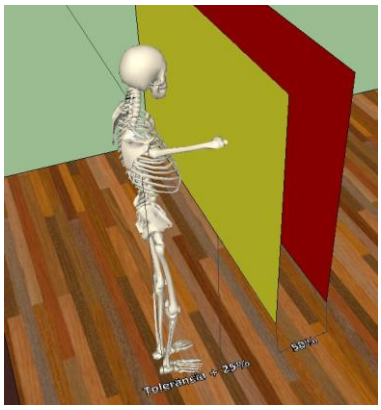


Figura 3 - Plano de escrita de duplo limite

O segmento seguinte é o maior dos três segmentos. Vai dos 25% aos 75% e é através deste que o utilizador deverá interagir maioritariamente, por se revelar essencial na movimentação e interacção com a aplicação. Sempre que o braço seja movimentado dentro destes limites, as coordenadas bidimensionais *(x, y)* da mão serão transpostas para o ecrã, movimentando o cursor também segundo as mesmas coordenadas. Neste segmento, os movimentos da mão do utilizador não se traduzem em desenhos no ecrã.

O último segmento vai dos 75% aos 100%. Nele, tal como no segmento anterior, é feita a transposição das coordenadas da mão para o cursor no ecrã. Neste caso, os gestos esboçados pelo utilizador são exibidos no ecrã.

Figura 4 - Utilizador dextro, com a mão projectada, prestes a ir de encontro ao limite estipulado para desenho

De seguida é necessário identificar qual das mãos será usada para desenhar as letras. Este processo foi pensado de forma a não limitar a interacção exclusivamente a utilizadores dextros, mas também ter em consideração o

público sinistro. Assim, o utilizador deverá projectar para a frente a mão que pretende usar de modo a interceptar o segmento de desenho (75%) como se encontra exemplificado na Figura 12.

Dos três algoritmos implementados, dois deles, o KDA e o SVM, recorrem a um método de classificação. Já o $N recorre a trigonometria simples, o que o torna mais rápido no processo de carregamento inicial das amostras em comparação com os dois anteriores. O $N apoia a sua análise num conjunto de 41 amostras, isto é, um número superior ao das 26 letras que constituem o alfabeto. Isto deve-se ao facto de cada uma das letras poder ser desenhada de diferentes formas, iniciando o traço de diferentes posições e variando o número de traços usados de acordo com a forma como a letra é traçada. Por exemplo, a letra 'A' pode ser desenhada com três traços ou simplesmente usando apenas dois.

Uma vez terminado um desenho, este é analisado pelo reconhecedor $N, sendo indicado o resultado da identificação e apresentado o padrão que mais se assemelha de entre o conjunto de 41 amostras de aprendizagem usado.

1. VALIDAÇÕES

De modo a verificar a viabilidade da aplicação desenvolvida no âmbito deste projecto, procedeu-se a uma avaliação informal com um grupo de 20 participantes com idades compreendidas entre os 6 e os 25 anos, dos quais 7 elementos eram adultos, e os restantes crianças do ensino básico, tendo cada um deles realizado 10 tentativas. Ao longo dos testes foi possível verificar, por parte dos utilizadores, um certo embaraço nas duas tentativas iniciais, vindo progressivamente a adaptarem-se à interface ao longo das várias experiências realizadas. Considera-se positiva a adaptabilidade do utilizador à aplicação. No entanto, revelou-se essencial sujeitar o utilizador a um processo de aprendizagem. Ao longo das 10 tentativas, o utilizador vai-se familiarizando com os diferentes planos, adquirindo a prática necessária ao desenho das letras. Porém, após apenas três interacções, o utilizador ainda demonstra debilidades relacionadas com a sensibilidade na transposição dos gestos, bem como dificuldades em situações em que se pretende remover a mão do desenho sem deixar rastos indesejados no ecrã. Estas situações carecem de um treino mais profundo para serem gradualmente corrigidas, melhorando consideravelmente a interacção com o utilizador.

Relativamente ao reconhecedor de padrões eleito para a aplicação, o $N, a definição de amostras com diferentes ordens de desenho das 26 letras do alfabeto revelou-se essencial para a precisão no reconhecimento, obtendo este algoritmo uma taxa de sucesso de 92%, em comparação com os outros reconhecedores de padrões Kernel Discriminant Analysis (KDA, 63%) e Support Vector Machine (SVM, 54%) testados (Figura 13).

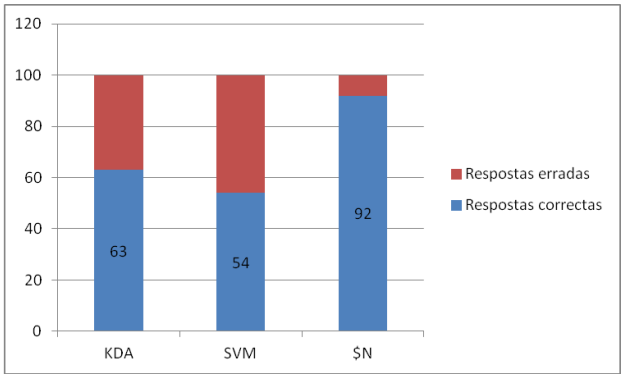


Figura 1 - Grau de eficiência dos algoritmos de reconhecimento de padrões testados

2. CONCLUSÃO E TRABALHO FUTURO

O estudo deste sistema de reconhecimento de gestos no ar, tem vindo a permitir conceber uma solução direccionada às crianças do ensino básico de forma a auxiliá-las, na aprendizagem da escrita. Além disso, os desenhos produzidos são posteriormente analisados pela aplicação com uma taxa de reconhecimento bastante positiva, nas identificações efectuadas pelo reconhecedor. Até ao momento, com base nos testes realizados em dois grupos de pessoas de diferentes faixas etárias, foi possível constatar as valias desta aplicação na área da educação e desenvolvimento motor das crianças, assim como diferentes níveis da percepção espacial, por parte dos utilizadores das diferentes faixas etárias. Apesar de os resultados obtidos indicarem que o público-alvo mais jovem demonstra uma maior dificuldade ao longo do processo de familiarização com os planos, em comparação com os utilizadores de idade mais avançada. Este público mais jovem demonstra, em contrapartida, uma maior capacidade de adaptação, melhorando gradualmente as suas capacidades de coordenação.

Um aspecto essencial, a considerar para os utilizadores de qualquer faixa etária, prende-se com a necessária familiarização aos planos de interacção, por forma a interagir eficientemente com a aplicação.

Como perspectivas de trabalho futuro, considera-se a utilização de um modelo dinâmico com um esqueleto interno, constituído por um conjunto de junções que façam correspondência com as junções captadas pelo Kinect/OpenNI, permitindo desta forma animar um modelo com os gestos realizados pelo utilizador.

3. REFERÊNCIAS

[Anthony and Wobbrock, 07a] Anthony L., Wobbrock J. $N Multistroke Recognizer in JavaScript. *University of Maryland - Baltimore and University of Washington* ,USA,2007.

[Anthony and Wobbrock, 07b] Anthony L., Wobbrock J. A Lightweight Multistroke Recognizer for User Interface Prototypes. *University of Maryland - Baltimore and University of Washington* ,USA,2007.

[Kolman and Margaliot, 09] Kolman E., Margaliot M. Knowledge-Based Neurocomputing: A Fuzzy Logic Approach, Springer, p. 1-12, 2009

[Maxwell, 12] Maxwell, Abril 2012.

<http://www.maxwell.lambda.ele.puc-rio.br/11498/11498_5.PDF>

[Mehlmann et al., 11] Mehlmann G., Kistler F., Endrass B., André E., Klimmt C., Wagner J., Smirra R. Analysis of Interactive Storytelling Applications, UK, 2011.

[OpenNI, 11a] OpenNI. Introducing OpenNI, Outubro 2011.

<http://75.98.78.94/images/stories/pdf/OpenNI_UserGuide_v4.pdf%20%5b1>

[OpenNI, 11b] OpenNI. ProgrammerGuide, Outubro 2011.

< http://openni.org/docs2/ProgrammerGuide.html>

[Pinto, 12b] Pinto P., 2012. SDK v1.5 do Kinect para Windows disponível para download, Maio 2012.

<http://pplware.sapo.pt/windows/software/sdk-v1-5-do-kinect-para-windows-disponvel-para-download/>

[Rogers, 11] Rogers R., 2011. Journal. Kinect with Linux. Houston.

[SideKick, 11] SideKick, , Outubro 2011.

< http://www.sidekick.co.il/>

[Sousa, 12a] Sousa C., 2010. - Kernel Support Vector Machines for Classification and Regression in C#, Abril 2012.

<http://crsouza.blogspot.pt/2010/04/kernel-support-vector-machines-for.html>

[Souza, 10b] Souza C.,2010. Handwriting Recognition Revisited: Kernel Support Vector Machines. The *Code Project*. Brazil.

[Sylverberg T., 07] Sylverberg T., Kristensson P., Leifler O., Berglund E., Drawing on Paper Maps: Reliable On-line Symbol Recognition of Handwritten Symbols Using a Digital Pen and a Mobile Phone, 2007.

[Tanaka et al., 12] Tanaka K., Parker J., Baradoy G., Sheehan D., Holash J, Katz L. A Comparison of Exergaming Interfaces for Use in Rehabilitation Programs and Research, Vol 6, 2012.

[Tscherrig, 11] Tscherrig J., 2011. Activity Recognition using Kinect.

[Willow Garage, 11] Willow Garage, Outubro 2011.

<http://www.willowgarage.com/blog?page=8>

[Wobbrock et al., 07a] Wobbrock J., Wilson A., Li Y. $1 Unistroke Recognizer in JavaScript. University of Washington and Microsoft Research, USA, 2007.

[Wobbrock et al., 07b] Wobbrock J., Wilson A., Li Y. Gestures without Libraries, Toolkits or Training:A $1 Recognizer for User Interface Prototypes. University of Washington and Microsoft Research, USA, 2007.

[Yongmin et al., 01] Yongmin L., Shaogang G., Liddell H.,Kernel Discriminant Analysis. University of London, *Department of Computer Science*, UK, 2001