Universidade de Aveiro Departamento de Engenharia Mecânica
2015

JOÃO DAVID MACEDO DO NASCIMENTO

Reconhecimento gestual por câmaras 3D para interação humano-computador.

Universidade de Aveiro Departamento de Engenharia Mecânica
2015

JOÃO DAVID MACEDO DO NASCIMENTO

Reconhecimento gestual por câmaras 3D para interação humano-computador.

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Mecânica, realizada sob a orientação científica do Engenheiro Abílio Manuel Ribeiro Borges, Assistente Convidado do Departamento de Engenharia Mecânica da Universidade de Aveiro

o júri

presidente	Prof. Doutor Carlos Alberto Moura Relvas Professor auxiliar, Universidade de Aveiro – Departamento de Engenharia Mecânica
arguente	Prof. Doutor Paulo Miguel de Jesus Dias Professor auxiliar, Universidade de Aveiro – Instituto de Engenharia Eletrónica e Telemática de Aveiro
orientador	Engenheiro Abílio Manuel Ribeiro Borges Assistente convidado, Universidade de Aveiro – Departamento de Engenharia Mecânica

agradecimentos

Em primeiro lugar agradeço ao meu orientador, o Engenheiro Abílio Borges pela disponibilidade para me assistir na realização desta dissertação. Por toda a motivação dada, não só durante a realização desta dissertação, como também em todas as unidades curriculares que me lecionou, em especial na disciplina de Sistemas de Visão e Perceção Industrial.
Um agradecimento especial à minha família, que tanto se sacrificaram para que fosse possível que eu e o meu irmão tivéssemos uma educação adequada. Por fim deixo um obrigado aos meus colegas e amigos, tanto os de Aveiro como os de Braga, e mais recentemente aos meus novos colegas na Atena.

palavras-chave

interação humano-computado, reconhecimento gestual, visão artificial, visão por computador, câmara 3D, Kinect, MATLAB, apresentações interativas, microsoft developer toolkit, optical flow.

resumo

A interação entre um utilizador e um programa (software) é um assunto com destaque e importância desde a criação do computador pessoal. Normalmente a interação é feita através de um teclado e um rato. Nesta dissertação pretende-se explorar e descrever diferentes alternativas para reconhecimento, deteção e identificação de gestos das mãos, para posterior implementação de comandos em diferentes aplicações informáticas e quadros interativos.

Para reconhecimento de gestos, realizou-se o estudo da geometria da mão, identificação da mão esquerda e da mão direita, utilizando imagens em profundidade obtidas pela câmara Kinect. Para reconhecimento da cor da pele recorreu-se a imagens a cores utilizando um método probabilístico de reconhecimento de pele. Para manipulação e comando de diferentes aplicações informáticas, nomeadamente o PowerPoint, foram desenvolvidos diferentes algoritmos para o reconhecimento de movimentos e gestos da mão esquerda e direita em tempo real.

keywords

gesture recognition, human-computer interaction, artificial vision, computer vision, 3D camera, Kinect, MATLAB, interactive presentations, microsoft developer toolkit, optical flow

abstract

The interaction between the user and the program (software) it's been a subject of extreme importance since the creation of the personal computer. Normally the interaction it's made using a keyboard and a mouse. This thesis aims to explore and describe different alternatives for recognition, detection and identification of hand gestures, so it can be implemented in commands for a diverse number of informatics applications as also interactive whiteboards.

For gestures recognition, it was made the study of the hand geometry, the identification of the right hand and the left hand, using depth images obtained by the Kinect camera. For the skin color recognition it was used a probabilistic method. For the manipulation and command of different informatics applications, like PowerPoint, were developed different algorithms for the recognition of movements and gestures by right and left hand in real time processing.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Equações

Capítulo 1

Introdução

Desde a criação do computador pessoal que a interação entre o utilizador e o computador foi um assunto de extrema importância, capaz de colocar as empresas com as interfaces mais desenvolvidas na vanguarda da tecnologia, dando-lhes uma vantagem em relação à concorrência. Os primeiros computadores pessoais tinham um ambiente muito rudimentar, não existia interface gráfica, tudo o que aparecia no ecrã era apenas texto verde em fundo preto [1]. O ambiente em linhas de comandos requeria apenas um teclado para que o utilizador pudesse dar as suas instruções [ver figura 1].

Mais tarde com a criação da interface gráfica, o rato tornou-se num instrumento indispensável para o utilizador transmitir os seus comandos à máquina, podendo assim transmitir as suas instruções de uma forma mais *user friendly* [ver figura 2] permitindo ao utilizador realizar com mais instruções do que um computador só com um teclado. Desde então que a interação humano-computador, com fins mais pessoais, é feita com base nestes dois *inputs* que perduram até ao dia de hoje.

Figura 1. IBM 1981. Adaptado de extremetech.com

Figura 2. Apple Macintosh 1984. Adaptado de Apple Computer, Inc.

Contudo nem todos os computadores são iguais e nem todos os programas são controlados de igual maneira. Com o aparecimento de novas tecnologias como os *Tablet*/*Smartphones* e as consolas de jogos, outras formas de interação tiveram de ser desenvolvidas.

Nas consolas de jogo, o utilizador usa um comando com botões e *joysticks* para dar as suas instruções á consola de jogo [ver figura 3]. Os comandos também têm sofrido várias

alterações ao longo do tempo, os jogos foram sofrendo evoluções e foram requerendo mais funcionalidades que os comandos não conseguiam oferecer. A necessidade de evoluir desses comandos visava facilitar a interação entre o jogador e o jogo.

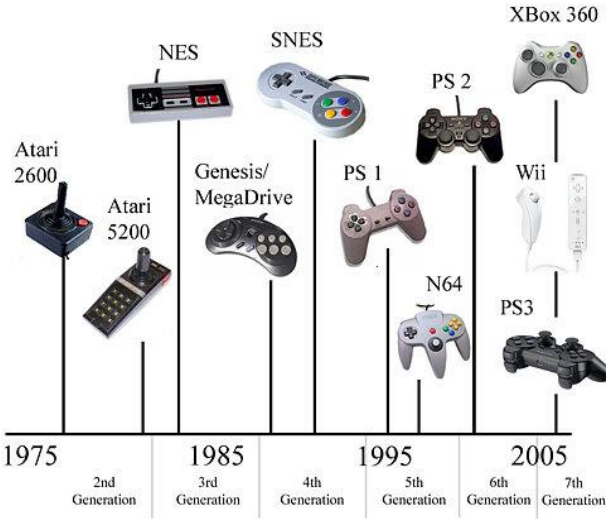


Figura 3. Evolução dos diversos comandos para consolas de jogos. Adaptado de [2]

Nos *Tablet* a interação é feita a partir de um ecrã táctil [ver figura 4].Apesar de se poder conectar um teclado e rato externos, o ecrã táctil permite a utilização do aparelho sem a necessidade da utilização de *inputs* mecânicos. .

Figura 4. Exemplo de um teclado embutido no ecrã tátil de um Tablet. Adaptado de [3]

Apesar de não existirem teclados e ratos físicos, o princípio de funcionamento é semelhante, o que prova que o teclado e o rato são componentes que muito dificilmente ficarão fora de moda.

Contudo, há cada vez mais oferta em relação à interação humano-computador. Atualmente grandes empresas de *software* apostam em assistentes virtuais como é o caso da Microsoft que apresenta a sua assistente Cortana, e da Apple que apresenta a Siri. Estas assistentes virtuais podem ser controladas a partir de comandos de voz, que, apesar de não serem novidade [4], cada vez se aposta mais nesta tecnologia.

A tecnologia de realidade virtual tem sofrido um rápido crescimento nos últimos anos. Cada vez mais empresas de topo apostam em equipamentos de baixo custo, permitindo ao público em geral uma nova forma de interação humano-computador, como é o caso do Microsoft Hololens [5] [ver figura 5], Samsung Gear VR [6] e etc…

Por fim e como tema desta dissertação, esta vai focar-se na interação humano-computador por sistemas de visão artificial. Inicialmente usado por companhias de videojogos como a Playstation com o Eye-Toy [7] [ver figura 8] e a Xbox com a Kinect [8] [ver figura 9], agora é-o cada vez mais em interações com programas de computador, como por exemplo o Google Earth.

Nesta dissertação, com recurso a uma câmara 3D Kinect, será feito o estudo da geometria da mão, como o reconhecimento de dedos e a distinção entre a mão direita da esquerda, análise de movimentos com recurso ao *Optical Flow*, bem como alguns gestos mais comuns e reconhecíveis para possíveis interações entre o utilizador e o computador, usando gestos como *input*, em vez da tradicional combinação entre o rato e o teclado.

1.1 Motivação

Apesar de atualmente o teclado e o rato estarem bem cimentados como pontes entre o utilizador e o computador, ainda há uma margem para evolução. Exemplo disso é a aposta que se faz atualmente na realidade virtual. Por vezes o teclado e o rato ficam limitados dependendo do objetivo da interação que pretendemos com o computador, como por exemplo as apresentações em *PowerPoint*. Estando num auditório a fazer uma apresentação em *PowerPoint* para uma audiência, por vezes torna-se necessário ao orador movimentar-se afastando-se do computador. Neste caso a utilização de um sistema de visão que esteja preparado para traduzir os gestos de um apresentador em comandos torna-se bastante útil . O apresentador pode fazer um gesto com a mão numa direção e o slide passa para o seguinte, o mesmo gesto mas na direção oposta, faz com que o slide retroceda; afastando as mãos uma da outra será possível fazer *Zoom in*. Existem vários comandos que poderiam ser traduzidos por gestos.

Libertar as mãos ao orador dará assim mais autonomia para este realizar outras tarefas.

Figura 5. Hololens da Microsoft. Adaptado de [5]

Figura 6. Representação do controlo de hologramas com recurso ao Hololens. Adaptado de [5]

Figura 7. Imagem do filme Relatório Minoritário, exemplificando a utilização de gestos para uma interação humano-computador. Créditos Twentieth Century Fox & Dreamworks

Para além do exemplo do *PowerPoint* outros *softwares* podiam ser controlados de forma semelhante, como por exemplo o Google Earth. Aplicações mais avançadas incluem videojogos, controlo de hologramas ou até mesmo quadros iterativos para salas de aulas.

Fazer o reconhecimento gestual é um grande desafio que envolve bons conhecimentos na área de visão artificial. Alguns dos desafios mais importantes deste trabalho podem resumir-se da seguinte forma:

- Segmentação das mãos de modo a que na imagem a ser processada aparecesse apenas as mãos e não o resto do corpo.
- Determinação da localização das pontas dos dedos e contagem do número de dedos levantados.
- Distinção da mão esquerda e da mão direita.
- Determinação do movimento que a mão está a realizar com recurso ao *Optical Flow* e utilização de filtros para uma obtenção de posições mais precisas.
- Determinação de alguns gestos comuns e sua aplicação em atuação no ambiente Windows.
- Atuação, com controlo do cursor do rato numa apresentação *PowerPoint*.

Contudo o maior desafio será combinar todos os objetivos acima listados num programa que fizesse o processamento do fluxo de vídeo em tempo real. Alguns dos objetivos podem ser feitos de várias maneiras. A escolha do método adequado para um processamento em tempo real, será de extrema importância.

1.2 Estrutura da dissertação

A dissertação está dividida em 4 capítulos. O capítulo 1 diz respeito à introdução da tese e a motivação, com a enumeração dos objetivos propostos.

No capítulo 2 teremos uma análise ao estado da arte em sistemas de visão para interação humano-computador já existentes.

O capítulo 3 será o desenvolvimento do programa, bem como os algoritmos escolhidos e metodologia usada. Neste capítulo está subdividido em desafios solucionados, como a segmentação das mãos, a determinação do número de dedos levantados, etc…

No último capítulo será feita a análise aos algoritmos desenvolvidos, a conclusão dos resultados obtidos, bem como uma apresentação de proposta para trabalho futuro.

Capítulo 2

Estado da arte

2.1 Câmara Kinect

A utilização de sistemas de visão artificial para interação humano-computador é uma tecnologia relativamente recente, historicamente mais associado à indústria cinematográfica [9] e à indústria de videojogos [10] sendo ainda pouco utilizada na área mais casual e não industrial como o caso dos computadores pessoais.

Na área dos videojogos o primeiro sistema de visão artificial com grande sucesso comercial foi criado em 2003. O sistema Eye-Toy da PlayStation permitia ao utilizador jogar certos jogos com recurso a gestos em vez do comando. A câmara usada era uma câmara compacta de baixa resolução e sem imagem de profundidade [ver figura 8].

Figura 8. Câmara Eye Toy da Sony. Créditos Sony Entertainment

Com o sucesso da Eye-Toy e sendo ainda um sistema muito rudimentar, a empresa Microsoft criou e lançou no mercado em 2009 a câmara Kinect [ver figura 9].

Figura 9. Câmara Kinect da Microsoft. Créditos Microsoft

A câmara Kinect foi desenvolvida principalmente para a consola de jogos XBOX 360. Com uma imagem *RGB* com mais resolução que a Eye-Toy, tinha como grande novidade a tecnologia de imagem em profundidade, desenvolvida pela empresa PrimeSense, esta tecnologia baseada na projeção de padrões de pontos infravermelhos [ver figura 12], permite com alguma resolução determinar a distância a que um objeto se encontra da câmara.

Atualmente existem duas versões da Kinect, a versão da XBOX 360 [ver figura 9] câmara que foi usada durante a dissertação, e a versão XBOX ONE [ver figura 10], uma evolução da Kinect da XBOX 360, com mais resolução na imagem a cores e um campo de visão mais amplo, para além de outras características como o reconhecimento por voz mais eficiente e a capacidade de fazer o reconhecimento e seguimento a 6 pessoas simultaneamente [11].

Figura 10. Kinect para a Xbox One. Créditos: xbox.com

Originalmente lançada para a indústria das consolas de vídeo jogos, a Kinect também pode ser usada com o computador pessoal. Com o lançamento da Kinect *SDK* (*Software Development Kit*), qualquer pessoa com uma câmara Kinect pode obter e tratar a imagem obtida no seu computador, o Kinect SDK atualmente na versão 2.0, vem com vários exemplos de código em *open source* (C++ e C#), incluindo a obtenção da imagem *RGB* e profundidade, o seguimento ao esqueleto de duas pessoas em simultâneo, reconstrução 3D e etc…

2.2 Princípio de funcionamento 3D da Kinect

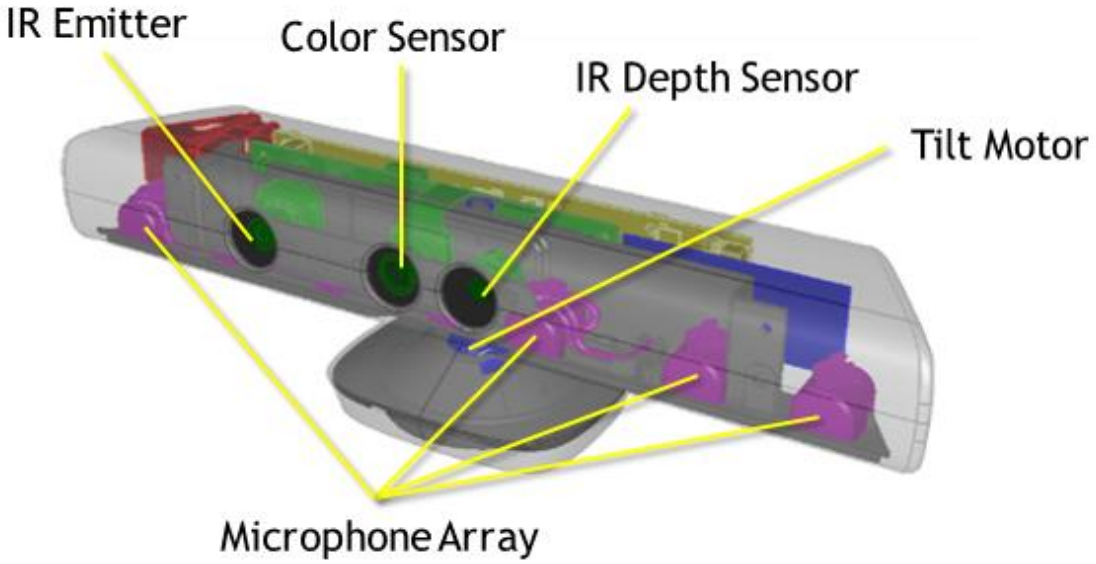


Figura 11. Representação dos componentes que fazem a câmara Kinect. Créditos: Microsoft

Como já foi referido acima, o princípio de funcionamento 3D da Kinect baseia-se na projeção de padrões de luz infravermelhos combinado com tecnologia *Stereo Vision*. A chamada *active triangulation* [12], em que em vez de usar duas câmaras, uma delas é substituída por um emissor de luz infravermelha.

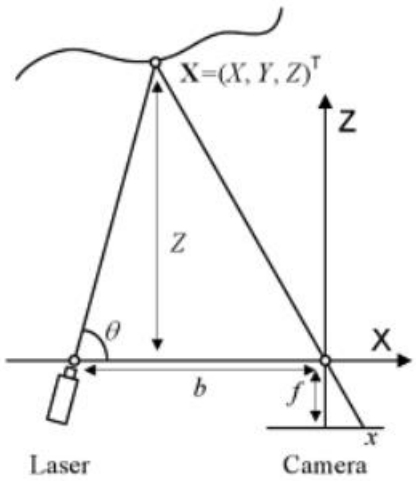


Figura 12. Active triangulation. Adaptado de [12]

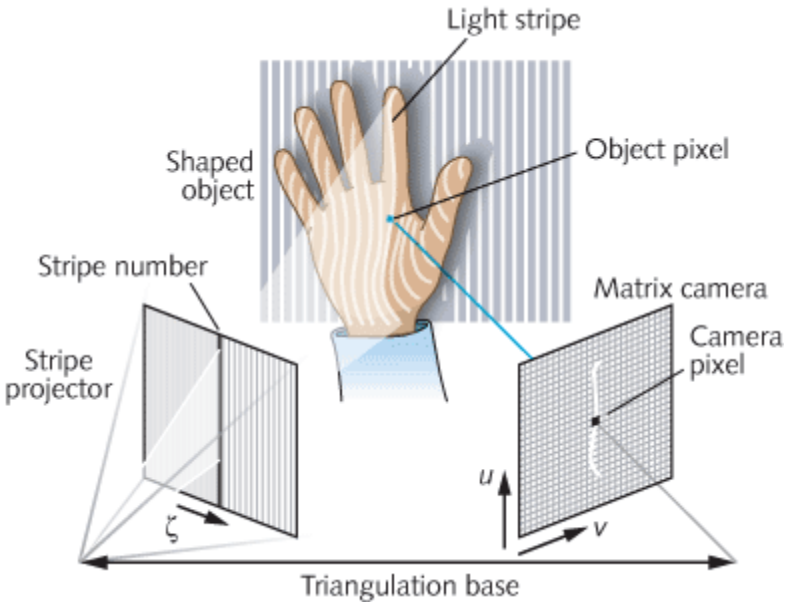


Figura 13. Exemplo de triangulação com recurso a Structured light. Neste caso a luz é emitida em forma de linhas verticais. Adaptado de [13]

A luz infravermelha projeta um conjunto de pontos com um certo padrão conhecido, ou seja, não aleatório [14] [ver figura 14].

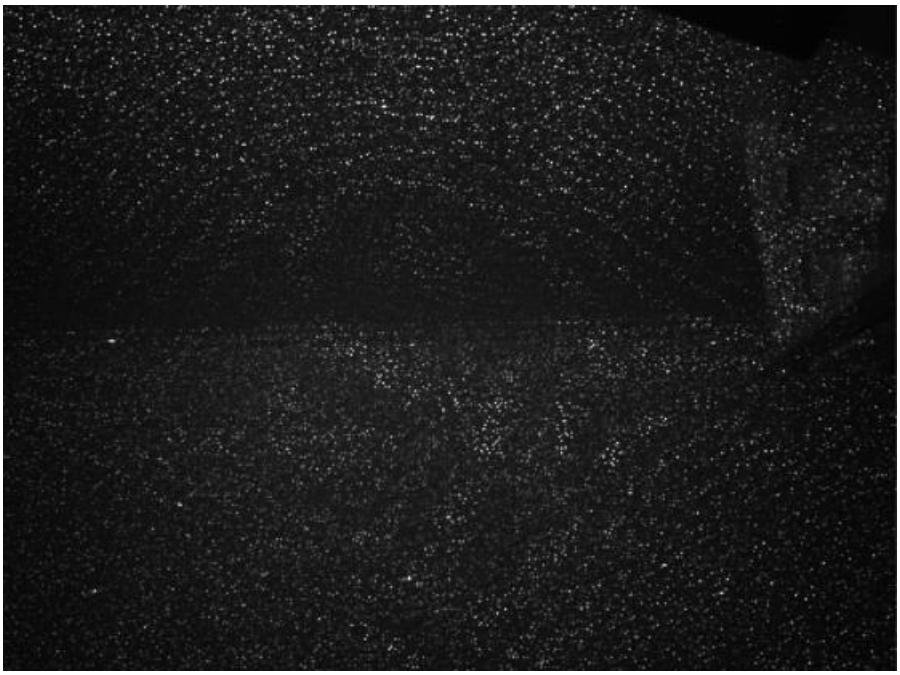


Figura 14. Pontos infravermelhos em forma de padrão obtido pela Kinect.

Este sistema tem grandes desvantagens: como depende muito de luz infravermelha, qualquer fonte de infravermelho exterior à Kinect vai causar interferências no padrão criado. Portanto, a Kinect não pode ser usada em ambientes exteriores ou com muita iluminação natural, além disso a utilização de duas Kinect tem interferências entre si.

A distorção a que o padrão de pontos está sujeito depende da distância a que este está projetado da câmara. Então, com recurso a esta informação, a medida da distância do padrão

projetado à câmara é feita com o princípio de câmaras stereo e com a informação de quanto mais longe o padrão estiver projetado, maior é a sua distorção [15]. A fonte emissora do padrão está a uma distância conhecida da fonte recetora [ver figura 15], com isso é possível saber a distância a que um objeto está da câmara.

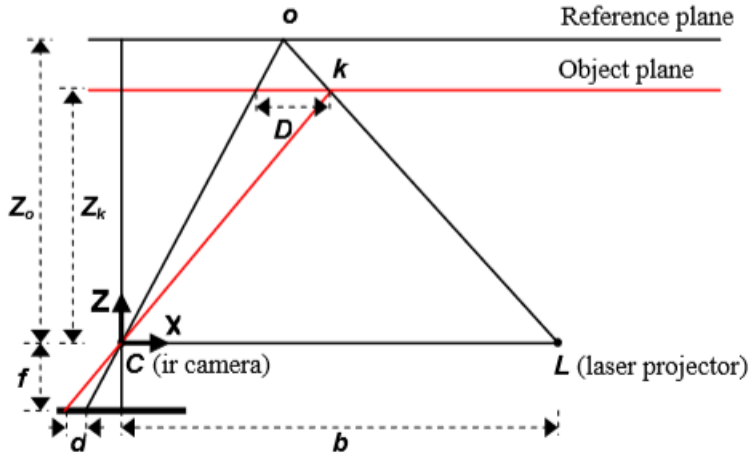


Figura 15. Representação da triangulação obtida. Adaptado de[15].

As equações que resolvem as distâncias pretendidas:

$$\frac{d}{f} = \frac{D}{Z_k}$$

Equação 1. Relação de semelhança entre triângulos. Adaptado de [15].

Substituindo D pela relação entre o *baseline* e o plano de referência

$$Z_k = \frac{Z_o}{1 + \frac{Z_o}{fb} d}$$

Equação 2. Determinação da distância em profundidade de um objeto à câmara. Adaptado de [15].

Conhecida a distância entre a câmara e o objeto, é possível também determinar as distâncias no plano xy a um ponto do objeto.

$$X_k = -\frac{Z_k}{f}(x_k - x_o + \delta_x)$$

Equação 3. Distância de um ponto ao centro da imagem no plano X. Adaptado de [15].

$$Y_k = -\frac{Z_k}{f}(y_k - y_o + \delta_y)$$

Equação 4. Distância de um ponto ao centro da imagem no plano Y. Adaptado de [15].

- Distância focal (f)
- b é o *baseline* (distância entre o recetor e o emissor de infravermelho)
- y_o e x_o são as coordenadas do ponto principal
- y_k e x_k são as coordenadas da imagem

- δ_y e δ_x são os coeficientes de distorção da lente

O princípio de funcionamento 3D da Kinect é extremamente importante, sendo importante perceber em que condições é possível obter uma imagem definida para o seu processamento. O conhecimento das limitações da Kinect ajuda a entender por que razão algumas das imagens obtidas durante a realização da dissertação são pouco definidas, e portanto, conduzem a um processamento da mão de forma errada.

Outro pormenor importante da Kinect são as suas limitações espaciais. A distância mínima à Kinect é de 800mm e a distância máxima 4000mm em modo *default*. Para obter a máxima definição da mão possível, a maior parte dos testes realizados durante a dissertação foram feitos entre os 800mm e os 1000mm da câmara.

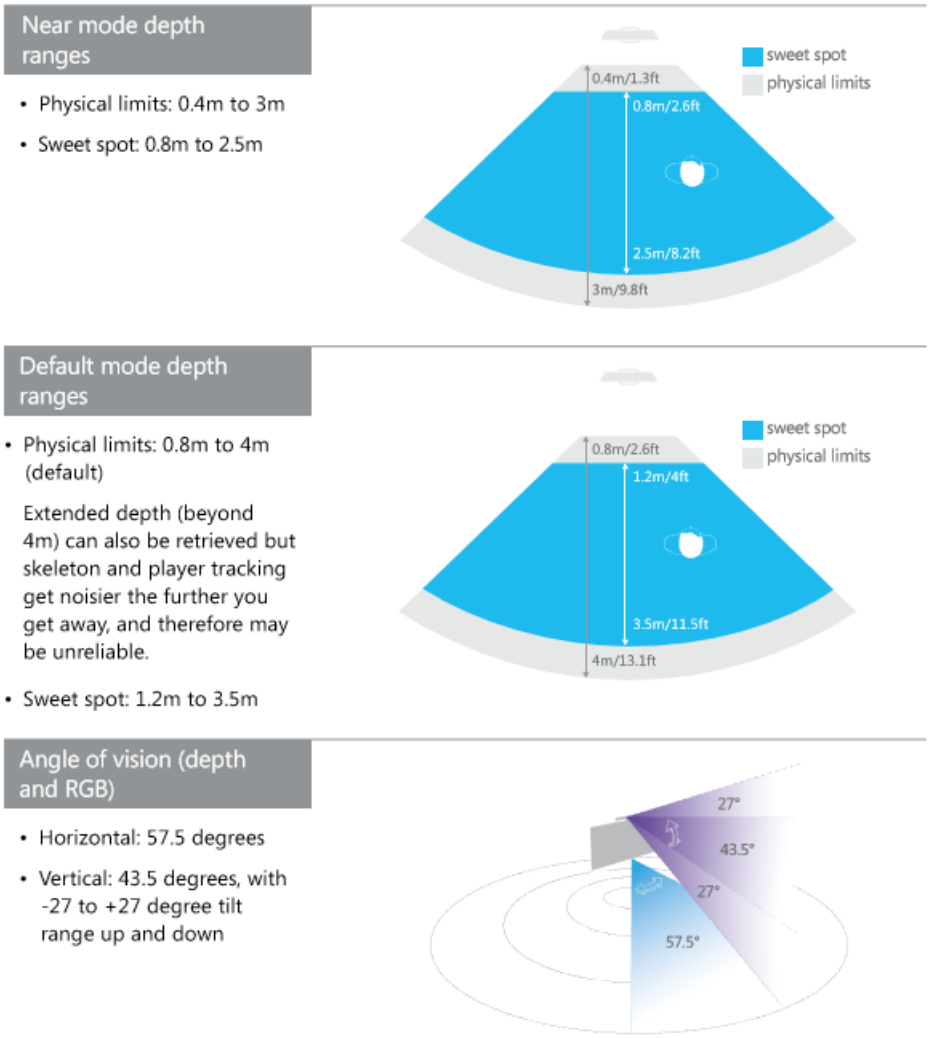


Figura 16. Limitações físicas da Kinect. Adaptado de [8]

2.3 Exemplos de aplicações com a Kinect

Como acima referido, a existência de uma câmara 3D fiável de baixo custo, com um *SDK*, deu a possibilidade de que qualquer pessoa pudesse criar programas para interação humano-computador, para além dos trabalhos desenvolvidos por empresas, também uma grande parte do trabalho é desenvolvido sem fins comerciais diretos disponíveis ao público em código *open source*.

Alguns dos exemplos encontrados durante a pesquisa bibliográfica:

A própria Microsoft com os jogos desenvolvidos para a XBOX:

Figura 17. Jogo para a Xbox com recurso à Kinect: Your Shape Fitness Evolved.

Figura 18. Jogo para a Xbox: Kinect Sports.

Figura 19. Jogo para a Xbox. Dance Central.

Os jogos representados acima baseiam-se na capacidade do *software* da Kinect de detetar e seguir o esqueleto dos utilizadores como visto na imagem abaixo [ver figura 20 e 21] [16].

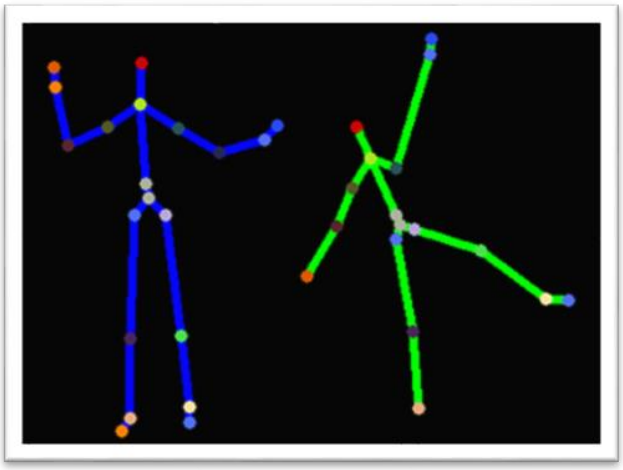


Figura 20. Seguimento do esqueleto de dois utilizadores.

A Kinect permite o seguimento de dois esqueletos simultaneamente, e o utilizador pode estar em pé ou sentado.

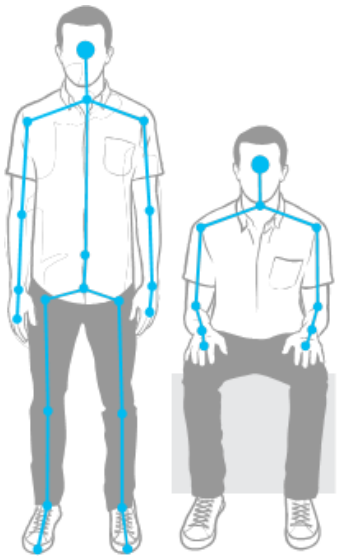


Figura 21. Representação do seguimento do esqueleto para utilizador em pé e sentado. Créditos Microsoft

Outras empresas criam e desenvolvem os seus programas com o intuito de os comercializarem. Como é o caso da Jintronix, que usa uma câmara Kinect para reabilitação motora.

Reabilitação/ fisioterapia, por parte da empresa Jintronix [17]

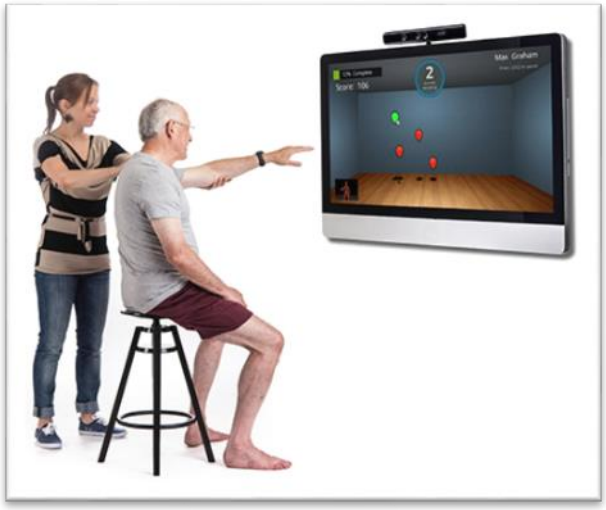


Figura 22. Aplicação da Kinect para reabilitação motora da Jintronix.

Aplicações para interação humano-computador como o projeto Kinect *Hands Recognition* em *open source* (criado para o sistema operativo Linux) desenvolvido pela *MIT* (*Massachusets Institute of Technology*), demos e os respetivos códigos fonte destes algoritmos desenvolvidos em combinação com o *PCL* (*Point Cloud Library*) e o *OpenNI*, encontram-se disponíveis no *ROS (Robot Operating System)* [18] [19]:

Figura 23. Interação Humano-Computador pelo MIT

Por fim uma aplicação já existente em interação humano-computador usando uma Kinect disponível para compra, pertence à empresa Evoluce, que oferece a possibilidade de controlar com gestos alguns programas mais usados:

- Windows 7
- Windows Media Center
- Microsoft Office
- Internet Explorer
- Etc..

Figura 24. Interação Humano-Computador pela empresa Evoluce.

Para além da utilização da Kinect, outras empresas usam outros tipos de sensores, que não recorrem a sistemas de visão.

O Google tem um projeto denominado como Soli em que o reconhecimento de gestos é feito com recurso a um radar. [20]

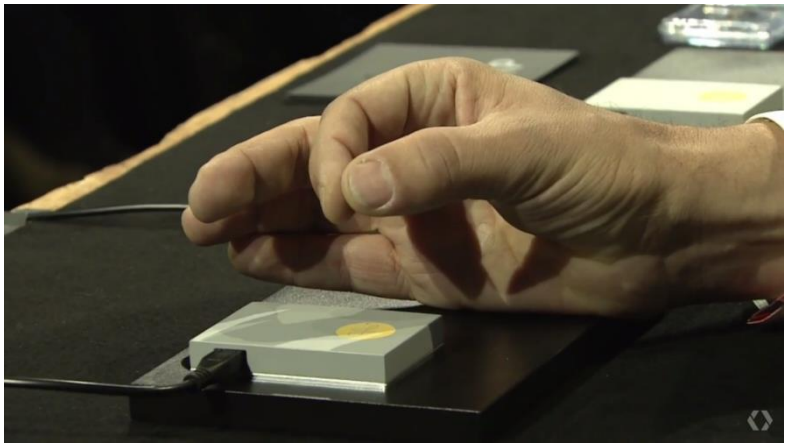


Figura 25. Radar do projeto Soli da Google. Créditos: Google.

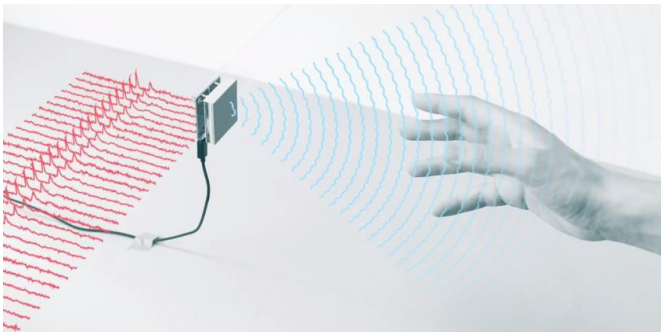


Figura 26. Ilustração do funcionamento do sistema de radar.

Outro sistema de reconhecimento de gestos sem recurso a câmaras existente, pertence à empresa Myo. Ao contrário dos sensores por visão e radar que podem ser utilizados à distância, a empresa Myo criou uma pulseira que deteta o movimento dos músculos. Com recurso a sensores EMG (Eletromiografia), é possível monitorizar a atividade elétrica produzida pelos músculos, nomeadamente a diferença de potencial entre dois, ou mais elétrodos [21], Aplicando filtros e algoritmos de análise de sinais é possível traduzir os sinais elétricos gerados pelos músculos em gestos/movimentos para comandos de atuação [ver figura 27]. A pulseira é não intrusiva, ou seja, não são necessárias agulhas que penetrem a pele, os sensores são colocados à superfície da pele. Para além dos sensores EMG a pulseira tem também acelerómetros e giroscópios.

Após a análise dos sinais pelos algoritmos da Myo é possível fazer a atuação em diversos programas, a comunicação entre a pulseira e o computador é feito por sinal Bluetooth.

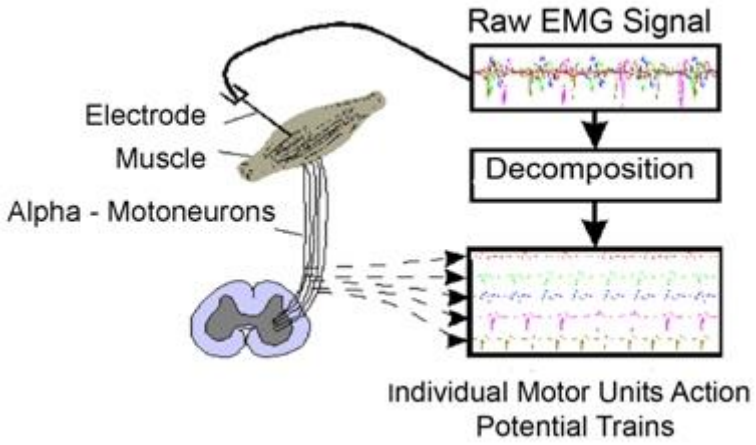


Figura 27. Representação do funcionamento do sensor EMG.

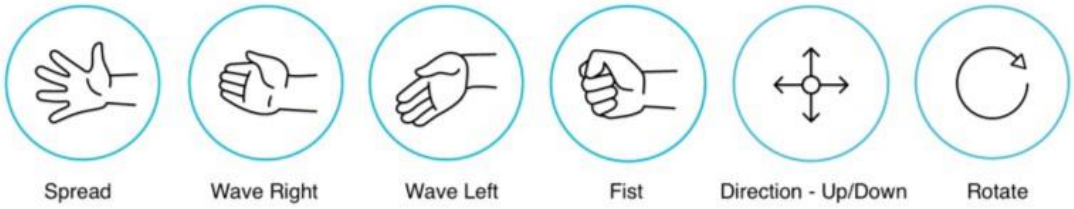


Figura 28. Alguns gestos detetados pelo sistema Myo.

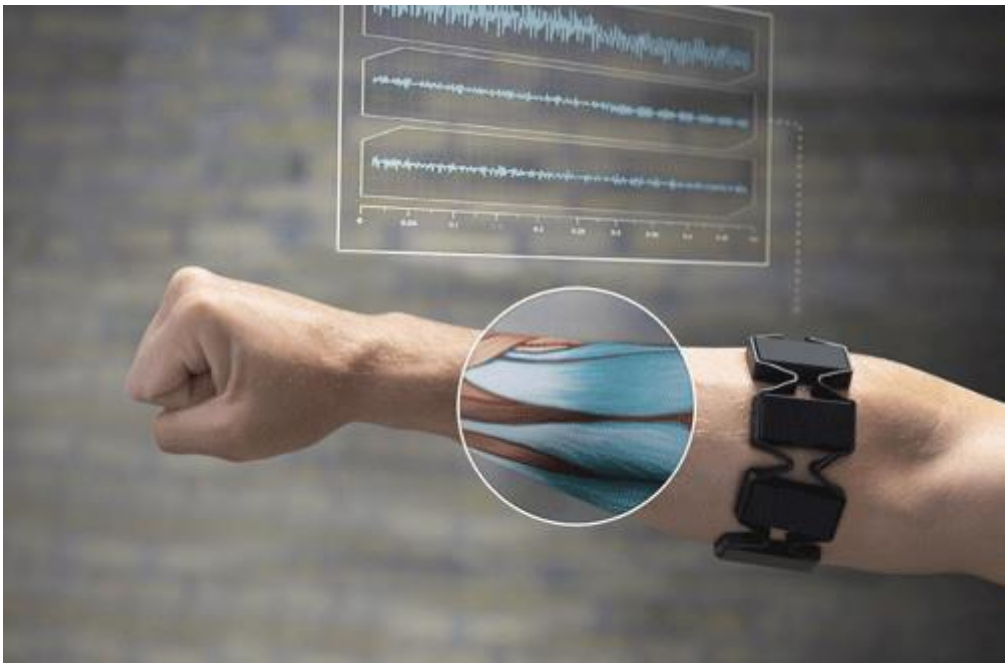


Figura 29. Pulseira Myo.

2.4 Técnicas existentes de reconhecimento

Atualmente existem alguns métodos para o reconhecimento de gestos e deteção de dedos. Existem algoritmos mais complexos como a utilização de librarias como a *Point Cloud Library*, para fazer a reconstrução 3D da mão, e outros que extraem a informação da imagem 3D para fazer uma representação 2D. Este último foi o método usado durante a dissertação. O estudo da mão será feito a partir de uma imagem binária 2D.

Começando com a segmentação das mãos, que pode ser feita por reconhecimento da cor da pele por extração 3D, a conjunção dos dois métodos ou também por subtração do fundo. Para a segmentação das mãos por cor da pele é feito um estudo dos espaços de cor existente, normalmente o mais usado o *YCbCr*, onde depois é feito o método probabilístico para determinar se o pixel em estudo pertence ou não à pele, com recurso a imagens de teste e o estudo de histogramas é preciso fazer uma análise probabilística para ignorar alguns pixéis fora de contexto como por exemplo falsos positivos ou falsos negativos. [22]

A segmentação feita por profundidade é um método que já vai requerer a utilização das capacidades de imagem 3D da Kinect ou de duas câmaras stereo. Na pesquisa bibliográfica a segmentação por profundidade é na sua maioria realizada com recurso à câmara Kinect.

O reconhecimento da mão em si envolve algoritmos e um raciocínio mais complexo. A forma da mão está em constante mudança. Alguns métodos usados para o reconhecimento de gestos são [23]:

- *Hidden Markov Models*
- *Finite State Machine/model matching*
- *Features Extrating*
- Estudo das propriedades geométricas da mão (Área, fator de forma, momentos invariantes)

Ao contrário da pesquisa bibliográfica, a determinação dos gestos da mão não foi feita por *template matching* mas sim pelo estudo das propriedades geométricas, com foco no fator de forma e nos momentos invariantes.

Para detetar e localizar os dedos, existem alguns procedimentos como por exemplo: a curvatura da ponta do dedo, com auxílio ao *Convex Hull* [24], ou o *Finger Earth Mover* [25].

No trabalho realizado estudou-se o ângulo da curvatura da ponta do dedo, recorrendo ao método *K-Curvature*.

Capítulo 3

Desenvolvimento

3.1 Plano de trabalho

O tema principal falado nesta dissertação é o estudo da mão para possíveis aplicações em ambientes interativos entre o utilizador e o computador. Temas como a segmentação das mãos, reconhecimento de gestos e contagem de dedos levantados e a análise de movimentos serão discutidos ao longo da dissertação.

O programa foi desenvolvido integralmente em MATLAB.

Para controlar o computador e alguns programas, o MATLAB não é a melhor opção. No entanto pode ter vantagem na análise de resultados obtidos. Apesar de terem sido conseguidos alguns dos objetivos de atuação, como o controlo do cursor do rato e a simulação do pressionar de teclas. Para o sistema operativo Windows, um programa em C++, com auxilio a librarias de processamento de imagem *OpenCV* e librarias de interação com o Windows como o *OpenNI*, seriam mais apropriadas relativamente ao MATLAB.

A capacidade de distinguir a mão esquerda da mão direita será um dos maiores desafios. Como as mãos podem estar em constante movimento e sempre a alterar a forma, a sua distinção torna-se bastante complexa. A capacidade de saber num dado instante que mão é que está presente para a câmara torna-se num desafio aliciante.

O estudo do movimento da mão com recurso a algoritmos de *Optical Flow*, também terá a sua importância e será discutido com cuidado durante a dissertação.

Apresenta-se seguidamente o esquema do trabalho a realizar:

3.2.1 Segmentação por cor da pele e profundidade

Para começar a estudar a forma da mão e para ter a possibilidade de usar a sua geometria e os seus movimentos em comandos interpretados pelo computador, é necessário dizer ao computador para procurar as mãos para poder trabalhar com elas.

Como foi referido anteriormente, apesar de ter as mãos apontadas para a câmara e de facto elas aparecerem na imagem, o computador, sem um pré-processamento, não irá distinguir as mãos dos outros objetos presentes na imagem, como por exemplo o corpo do utilizador. Portanto é preciso criar um algoritmo para fazer a separação das mãos do resto que aparece na imagem.

Como foi explicado no capítulo anterior, existem alguns procedimentos que permitem a separação das mãos do corpo do utilizador ou do fundo. Foram testados dois métodos para tentar obter a mão: a segmentação por cor da pele e a segmentação por profundidade.

A segmentação tanto pode ser feita por identificação dos pixéis que apresentem a cor da pele do utilizador, como pode ser feita por limitação de distância à câmara. A terceira opção seria uma combinação das imagens a cor e profundidade. Ao segmentar pela cor da pele há uma grande probabilidade de aparecer a cara do utilizador. Como a cara aparece atrás das mãos (do ponto de vista da câmara), pode-se eliminar a cara pela segmentação em profundidade. [ver figura 30]

Durante a realização desta dissertação a segmentação por cor de pele foi experimentada, aplicando o trabalho realizado pelo colega Rui Barbosa [22]. Usando a base de dados compilada com os valores de cores para várias tonalidades de pele, tentou-se obter as mãos para um processamento. Alguns resultados desses testes estão representados nas imagens abaixo.

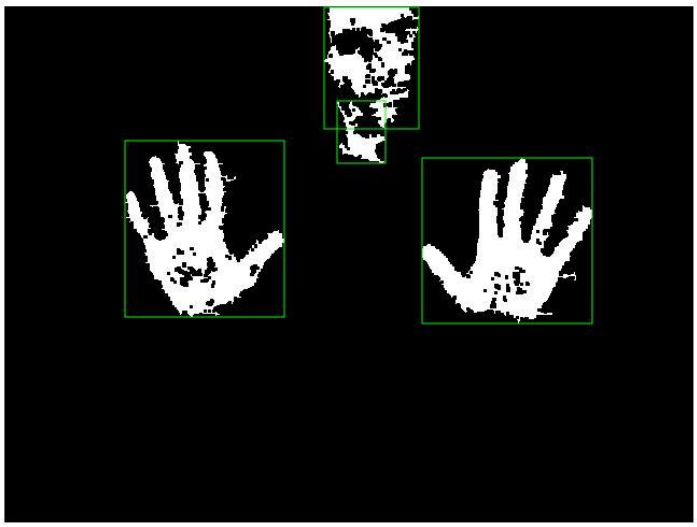


Figura 30. Segmentação por cor da pele.

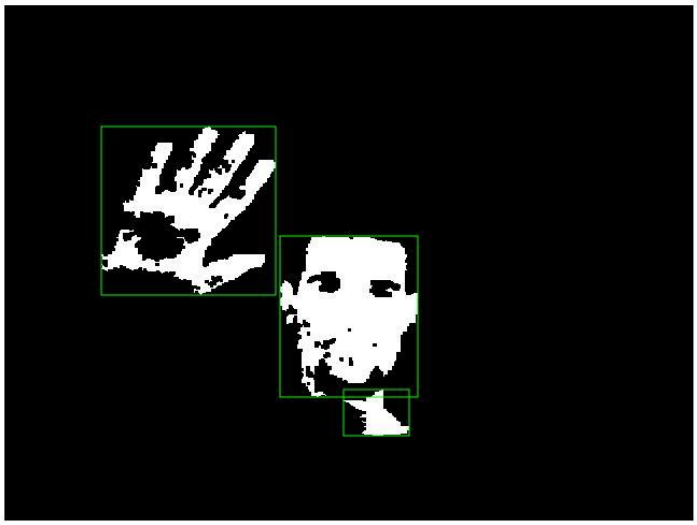


Figura 31. Segmentação por cor da pele.

Na figura acima verifica-se que a base de dados consegue detetar a cor da pele, apresentando a cara e a mão. No entanto, a imagem formada da mão não é perfeita e apresenta alguns buracos. Aplicando alguns algoritmos morfológicos seria possível eliminar pequenos buracos no meio da mão melhorando assim a sua forma. No entanto, em casos mais graves, existia uma grande parte da mão que não era detetada como demonstrado na figura abaixo [ver figura 32], tornando assim o processamento impossível. Apesar disso é de notar a excelente capacidade de eliminar todos os objetos incluindo o fundo e apresentar apenas as regiões que pertencem à pele.

Figura 32. Segmentação por cor da pele.

A utilização das cores para fazer a segmentação tem algumas limitações. Apesar do algoritmo ser bastante robusto não está imune a certas condições de luminosidade ou de tonalidades de pele mais brilhantes; quaisquer variações mais bruscas têm consequências negativas na obtenção de uma mão bem definida para o processamento. A incapacidade de conseguir imagens bens definidas para o processamento limita os resultados pretendidos. Apesar de ocasionalmente se conseguir obter imagens mais aceitáveis, a imprevisibilidade deste método torna-o pouco viável para uma interação humano-computador correta. Normalmente o reconhecimento de pele é usado para programas de reconhecimento facial, ou para detetar a presença de um humano [30].

No entanto, apesar da utilização da segmentação por cor só por si não ser o suficiente para a realização de uma segmentação aceitável, a sua combinação com a imagem em profundidade é útil para a obtenção da imagem binária bem definida.

Para que a imagem em profundidade dê os resultados em termos de segmentação pretendidos, são necessárias algumas regras:

- Como já foi referido no Estado da Arte, a câmara 3D da Kinect não pode ser utilizada no exterior ou em ambientes com muita luz natural.
- Não pode existir nenhum objeto no campo de visão entre as mãos e a câmara, ou seja, as mãos tem de ser o objeto mais próximo da câmara. Como também já referido no estado de arte as mãos não podem estar a menos de 800mm da câmara, e também não convém que estejam a mais do que 1200mm.

- A Kinect tem de estar a apontar para o utilizador, e este tem de estar de frente para a câmara, idealmente na perpendicular. A câmara pode estar em diferentes alturas mas as palmas da mão têm que estar perpendiculares com o eixo ótico [ver figura 33 e 34].

Figura 33. Exemplo de posição a assumir e da distância da Kinect, com as mãos a apontarem para a câmara.

Figura 34. Exemplificação da câmara a apontar para as mãos.

Com este conjunto de regras esclarecidas podemos usar as capacidades da câmara 3D em plenas condições para uma aquisição ideal da forma da mão.

Com estas condições definidas é possível estudar as imagens em profundidade obtidas pela câmara Kinect. Na imagem obtida, as profundidades estão ordenadas por escala em códigos de cores, objetos a azul estão mais próximos à câmara do que objetos a púrpura. [ver figura 35]

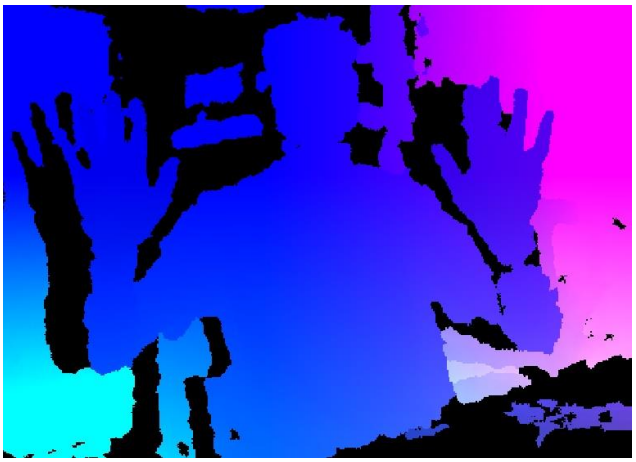


Figura 35. Imagem em profundidade da Kinect obtida pela Kinect SDK.

Obtendo os dados pela Kinect pode transformar-se a imagem em profundidade numa nuvem de pontos 3D, resultando numa análise mais intuitiva [ver figura 36]. Uma representação que, para além de estar num gráfico 3D de coordenadas [XYZ], se nota que quanto mais próximo estiver da câmara mais escura é a cor apresentada.

Figura 36. Nuvem de pontos obtida pela imagem em profundidade da Kinect dada pelo MATLAB.

Com a nuvem de pontos é possível aceder aos valores espaciais que se pretendem. A informação é guardada numa hipermatriz com 3 dimensões, em que cada matriz tem a dimensão de uma imagem de profundidade com 640x480 pixéis. A primeira matriz corresponde aos valores de distância em X, ou seja, visto na horizontal a partir do eixo ótico, a segunda matriz corresponde aos valores de Y, ou seja; visto na vertical a partir do eixo ótico, o ponto do eixo ótico tem como coordenada (X,Y) =(0,0); visto a partir do utilizador se o objeto estiver à direita do eixo ótico o valor de X <0, e se o objeto estiver acima do eixo ótico o valor de Y <0. A terceira dimensão da matriz corresponde à distância do objeto ao plano da imagem formado na câmara e será sempre Z> 0.

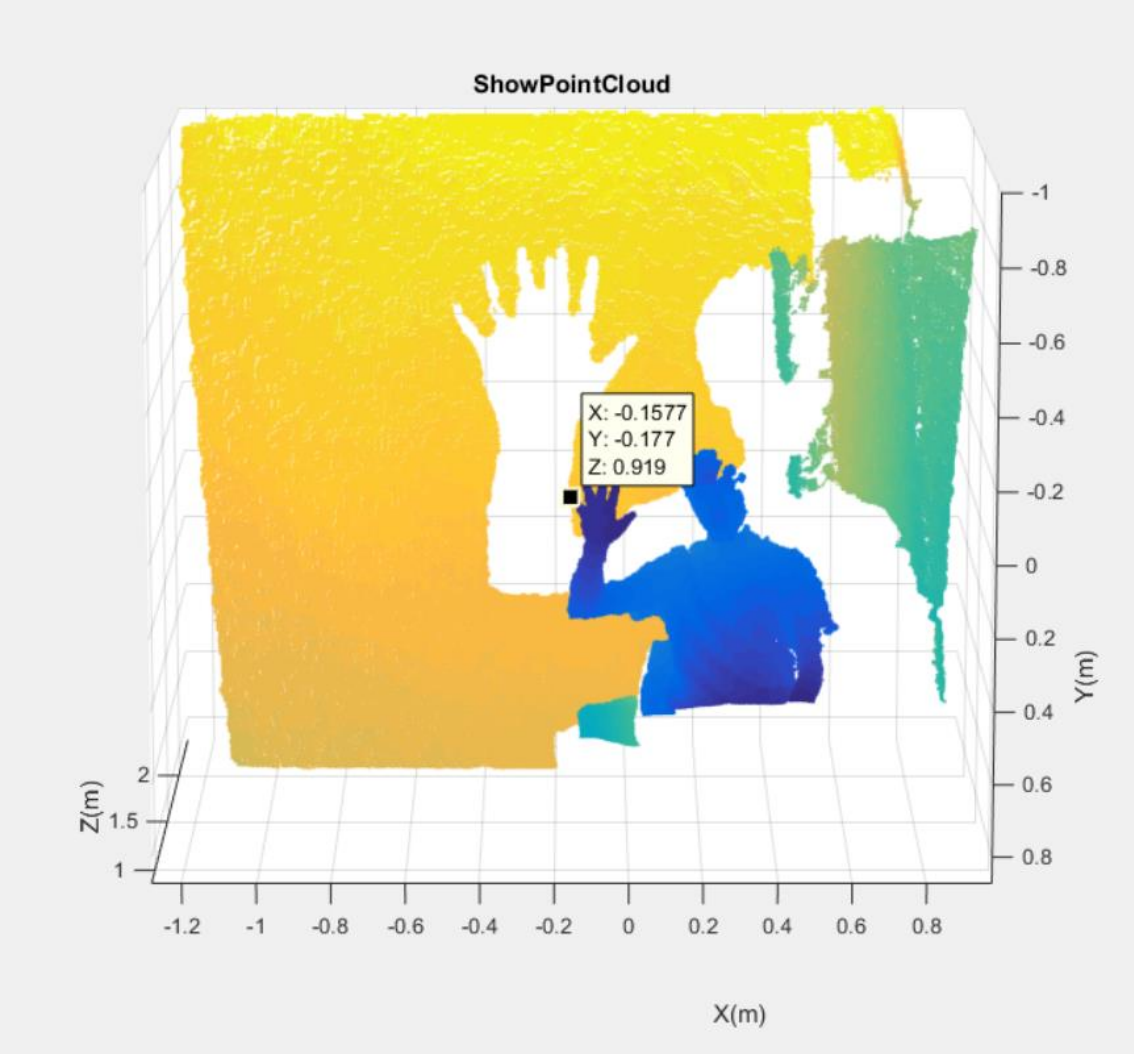


Figura 37. Exemplo das distâncias que se podem obter a partir da nuvem de pontos.

Com base na informação retirada das matrizes podemos saber sempre a que distância é que as mãos estão da câmara. Se as regras de utilização referidas acima forem aplicadas é possível com bastante facilidade obter a segmentação da mão.

Se as mãos forem o objeto mais próximo da câmara, retira-se o Z mínimo na terceira matriz dada pela nuvem de pontos. Para garantir que a mão, ou as duas mãos, aparecem separadas de tudo o resto dá-se um limite de profundidade.

$$Zona\ de\ Segmentação < Zmin + Lim$$

Equação 5. Limite da zona de segmentação.

Onde Zmin é a distância mais próxima à câmara, neste caso a mão, e Lim é um limite dado de profundidade para além da Profundidade mínima.

Caso o valor de Zmin seja inferior a 800mm ou superior a 1200mm o processamento da imagem é cancelado e o código passa para o frame seguinte.

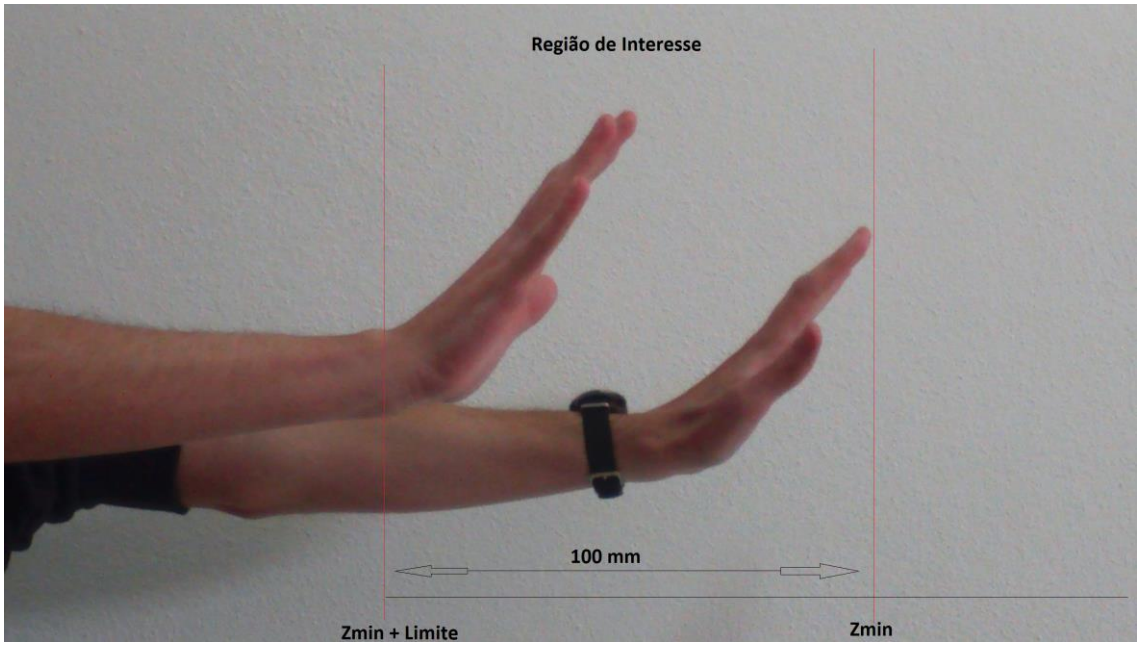


Figura 38. Demonstração dos Limites de segmentação.

Após a aplicação da segmentação, o resultado é uma imagem lógica 2D, em que os objetos a branco representam os objetos que cumprem os requisitos dos limites de profundidade impostos. Apesar de se perder as propriedades 3D que a nuvem de pontos oferecia, para o estudo da forma da mão, uma imagem binária é mais fácil de processar.

Apesar da imagem obtida por profundidade não ser perfeita, muito devido à resolução, os resultados são significativamente melhores do que os obtidos pela segmentação por cores.

Figura 39. Segmentação das mãos usando imagem em profundidade

Contudo com uma utilização mais correta e assim uma segmentação mais aceitável, os valores de Lim podem estar compreendidos entre 100mm e 200mm, (valores retirados experimentalmente). Valores abaixo de 100mm podem resultar em mãos incompletas. Caso uma das mãos esteja muito à frente da outra, a que fica atrás não será visualizada na imagem binária. Valores muito acima de 200mm podem apresentar outros objetos como por exemplo o

corpo ou a face do utilizador. Sendo assim é possível ao utilizador ter as duas mãos em planos com diferentes distâncias à câmara, desde que uma mão não esteja a mais de 200mm da outra.

Apesar de serem impostas diversas regras para uma utilização correta, se não houver um utilizador, ou na imagem apanhar, por exemplo, o canto de uma mesa entre a lente e o utilizador, o programa deixa de funcionar como é desejado. Para minimizar esses erros entra em ação a utilização da segmentação por cor.

Portanto, aplicando a segmentação por cor aliada à segmentação por profundidade os resultados melhoram significativamente.

O estudo da imagem a cores entra no código em primeiro e vai determinar as regiões com pixéis que podem pertencer à cor da pele.

A segmentação por cor vai estudar os valores da cor de cada pixel e sempre que a probabilidade de um pixel se encontrar dentro do limite do que pode ser considerado cor da pele, vai indicar que há de facto pele numa certa região da imagem. Agora que a pele foi encontrada numa região da imagem, a procura pela mão com recurso à utilização da imagem em profundidade inicia. Com isto evita-se que o programa esteja a correr em contínuo na esperança de encontrar uma mão, mesmo que não exista um utilizador na imagem.

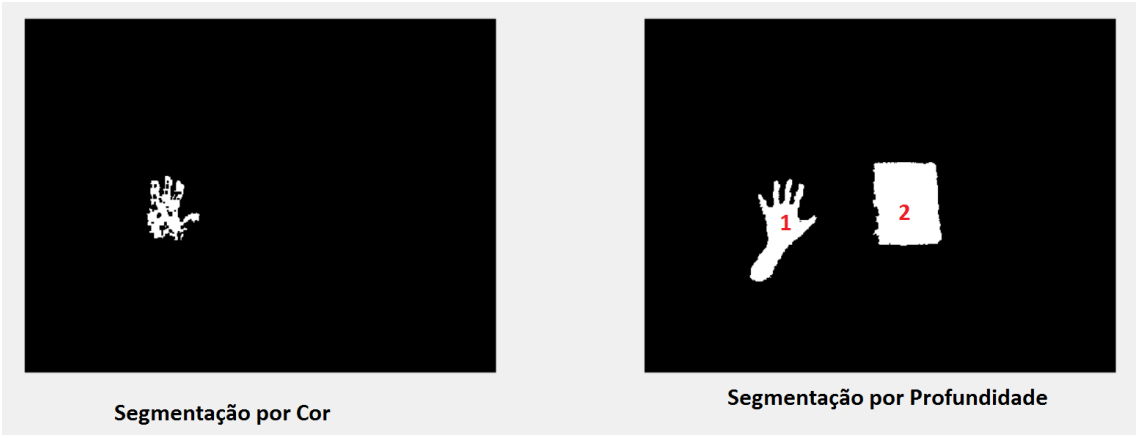


Figura 40. Representação do funcionamento da segmentação por cor da pele aliada à segmentação por profundidade.

Acima, na figura da direita, está um exemplo de uma mão e um objeto desconhecido, a partir da segmentação por cor de pele, na figura da esquerda é possível saber que existe um objeto de interesse, neste caso é uma mão, mas pode ser uma cara, no entanto para o efeito vamos apenas estudar o caso de uma mão e um objeto desconhecido. No entanto, a imagem em profundidade vai detetar a existência da mão (1) e de um objeto retangular (2) que não tem a cor da pele. Apesar de não dar para ter noção a partir das imagens demonstradas aquando da foto, o objeto 2 estava mais próximo da câmara do que o objeto 1. Então para estudar apenas o objeto 1, retiram-se as coordenadas na figura esquerda de todos os pontos do objeto mão, e na imagem em profundidade vai-se procurar a distância à câmara a que o objeto 1 está, visto que as coordenadas do objeto mão da figura da esquerda correspondem às coordenadas

dos pontos do objeto 1 na figura da esquerda. Deve ter-se em atenção que as imagens da Kinect a cores e em profundidade apresentam um *offset* que entretanto foi corrigido.

Assim a combinação das duas imagens minimiza a probabilidade de o programa estar a procurar por mãos quando elas não existem, e sabe-se sempre a distância a que um objeto de cor de pele está da câmara.

A partir das distâncias obtidas segmenta-se sempre pelo valor em Z mais próximo à câmara com a adição de uma margem, como já foi referido anteriormente, mas desta vez não procura o objeto mais próximo à câmara mas sim o objeto com a cor de pele mais próximo à câmara.

Para tentar simplificar a utilização das duas imagens pensou-se na utilização de operações binárias como a junção e a interseção, mas sem resultados convincentes. A utilização de interseção resulta apenas na imagem que se obtém na segmentação por cor, ou seja, a pior.

A junção das duas imagens dá o resultado da imagem em profundidade, por sinal a melhor que é possível obter das segmentações. Isto porque nos testes realizados a imagem em profundidade, apesar de poder apresentar falhas, costuma ser mais bem definida que a imagem segmentada pela cor.

3.2.2 Separação do antebraço

Mesmo assim há outra situação que precisa de um pré tratamento. Dependendo como o utilizador posiciona os braços em relação à câmara, estes podem aparecer na imagem binária, o que causa um transtorno. Para eliminar esta situação o utilizador tem de colocar as mãos em frente aos braços, ou seja, com os braços esticados, o que pode causar algum desconforto caso o utilizador esteja durante muito tempo com os braços esticados. Ao limitar a área máxima do objeto presente na imagem, pode-se eliminar o aparecimento do antebraço.

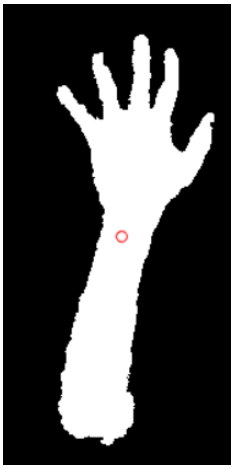


Figura 41. Imagem da mão com o antebraço obtida pela segmentação em profundidade, e representação da localização do centroide do objeto.

Na figura 41 vemos o resultado da segmentação quando aparece o braço. O aparecimento do antebraço na imagem complica o processamento. O centroide, devido ao efeito do antebraço aparece muito em baixo, neste caso na zona do pulso. O ideal seria aparecer no centro da palma da mão. Mas, dependendo da forma da mão, aberta ou fechada e da quantidade de braço que aparece, a posição do centroide variará bastante.

Neste caso o cálculo do centroide torna-se inútil para separar a mão do braço. Não devemos esquecer que a mão e o braço podem assumir várias posições. A imagem mostra o braço numa posição mais comum, na vertical, mas podem ocorrer situações em que o braço e a mão estejam na horizontal.

Limitar uma região de interesse (ROI) torna-se fundamental, e o centro da palma da mão é o local mais lógico para iniciar essa região. Com o centro da palma da mão é possível criar um círculo com determinado diâmetro que envolva a mão e a separe esta do antebraço, a partir de uma operação lógica.

Tudo o que aparecer dentro do círculo criado é separado para processamento. Sendo assim o resto do antebraço é eliminado.

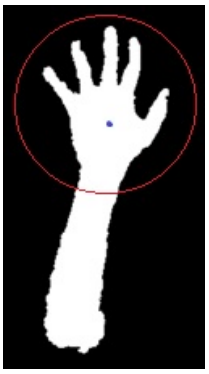


Figura 42. Representação desejada do centroide e circulo a limitar a região de interesse.

A solução encontrada para resolver este problema foi baseada no trabalho "*Hand tracking and gesture recognition system for human-computer interaction using low-cost hardware*" [24]. O princípio de aplicação é bastante simples, mas infelizmente é um processo muito pesado em termos de cálculos, o que vai tornar o tempo de processamento da imagem muito mais lento.

Com a utilização de imagens binárias é possível aplicar um algoritmo que permite a obtenção das coordenadas dos pontos que perfazem a fronteira do objeto, neste caso a mão [26]. As coordenadas de todos os pontos fronteira do objeto tanto pode ser a mão, como a mão com o antebraço, são guardadas numa matriz em que o número de linhas corresponde ao número de pontos totais da fronteira e as colunas correspondem às coordenadas XY do ponto.

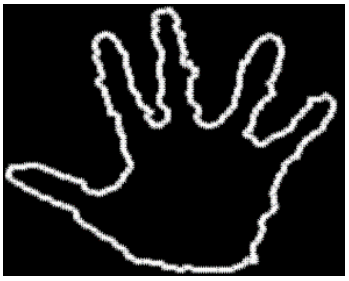


Figura 43. Obtenção da fronteira da mão.

Depois de determinados os pontos e as coordenadas pertencentes à fronteira, é necessário determinar todos os pontos pertencentes ao objeto.

Mais uma vez e aplicando o princípio anterior, todas as coordenadas são guardadas numa matriz em que o número de linhas corresponde aos pontos, e as colunas as coordenadas XY desses pontos.

Com esta informação guardada em duas matrizes, agora é preciso determinar a distância de um ponto pertencente à mão a todos os pontos que perfazem a fronteira da mão.

O ponto interior da mão, cuja distância mínima à fronteira tiver o valor mais elevado, é o ponto central da palma da mão. Na figura 44 temos uma representação de um ponto central da palma da mão, em que a distância mais pequena à fronteira é a mais alta de todos os outros pontos pertencentes ao interior da mão.

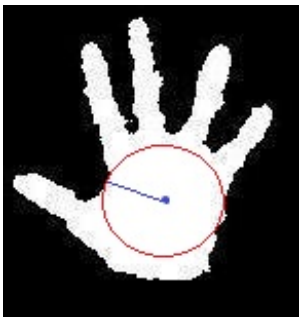


Figura 44. Exemplificação do ponto central da palma da mão, e do maior circulo que se pode ter dentro da mão.

Determina-se a distância euclidiana de cada um dos pontos interiores a todos os pontos da fronteira.

Os valores de distância são guardados em matrizes, em que cada matriz corresponde a um ponto interior da mão, e no interior da matriz temos as distâncias a todos os pontos correspondentes à fronteira. Ou seja, vamos ter um número de matrizes correspondentes ao número de pontos interiores, e a dimensão da matriz terá o número de colunas correspondentes ao número de pontos fronteiros existentes.

Depois de obtidas todas as distâncias teremos a distância mínima de cada matriz, ficando agora com um número de matrizes correspondente ao mesmo número de pontos interiores. Agora cada matriz tem apenas um valor; por fim determina-se qual dessas matrizes tem o valor mais alto.

A matriz restante contém as coordenadas do ponto, o mais afastado possível da fronteira.

Apesar dos resultados serem excelentes na determinação do centro da palma da mão, este método necessita de uma grande capacidade de processamento, requerendo assim muito tempo para fazer o processamento de um frame.

O número de cálculos realizados para obter este número de distâncias é bastante elevado. Por exemplo para uma imagem com 648 pontos de fronteira e 7543 pontos interiores dá um total de 4887864 distâncias calculadas, usando um poder de processamento de um Intel® Core™ i7 4700HQ 2.4GHz, e uma NVIDIA GeForce GTX 860M o tempo necessário para calcular esses valores foi de aproximadamente 440 segundos.

Figura 45. Determinação do centro da palma da mão fazendo o cálculo para todos os pontos.

O tempo necessário para o processamento da imagem é inaceitável, basta ter em atenção que este processamento foi apenas para a imagem de uma mão já pré tratada (não aparece o antebraço), e apenas uma mão, não duas. Se o tamanho da outra mão fosse semelhante, o tempo total de análise de um frame com duas mãos passava para o dobro.

Para tentar minimizar o tempo de processamento é necessário reduzir o número de distâncias a calcular. A partir de resultados experimentais, foi-se reduzindo o número de pontos a serem analisados, quer pontos interiores quer pontos fronteiros. Em vez de analisar os pontos um a um analisou-se de n em n pontos para os pontos interiores e de m em m pontos para os pontos fronteiros.

Tabela 1. Variação da posição do pixel central e do tempo decorrido de cálculo em função do número de pontos analisados.

Intervalo de pontos interiores	Intervalo de pontos fronteiros	Distância do ponto encontrado em relação ao ponto calculado para todos os pontos (pixéis)	Tempo de processamento (segundos)
1	1	-----	440.0
25	1	5	8.9

25	5	5	1.6
25	10	9.4	0.8
50	1	3.2	4.2
50	5	3.2	0.8
50	10	3.2	0.4
75	1	3.2	2.8
75	5	3.2	0.5
75	10	3.2	0.3
100	1	4.5	2.1
100	5	4.5	0.4
100	10	13	0.2
1	5	1	89.8
1	25	5	16.9
1	50	4	7.2

Apesar de esta ser uma melhoria significativa, em termos de processamento em tempo real, continua a ser muito tempo o que resulta em baixos frames por segundo Analisar movimentos com poucos frames por segundo torna-se muito complicado, e movimentos mais rápidos não são detetados.

Depois de feita a análise a uma mão, o mesmo processo pode ser aplicado para uma segunda mão.

Cada mão é processada separadamente e sequencialmente, portanto o processamento da segunda mão terá que esperar que a primeira mão acabe de ser processada.

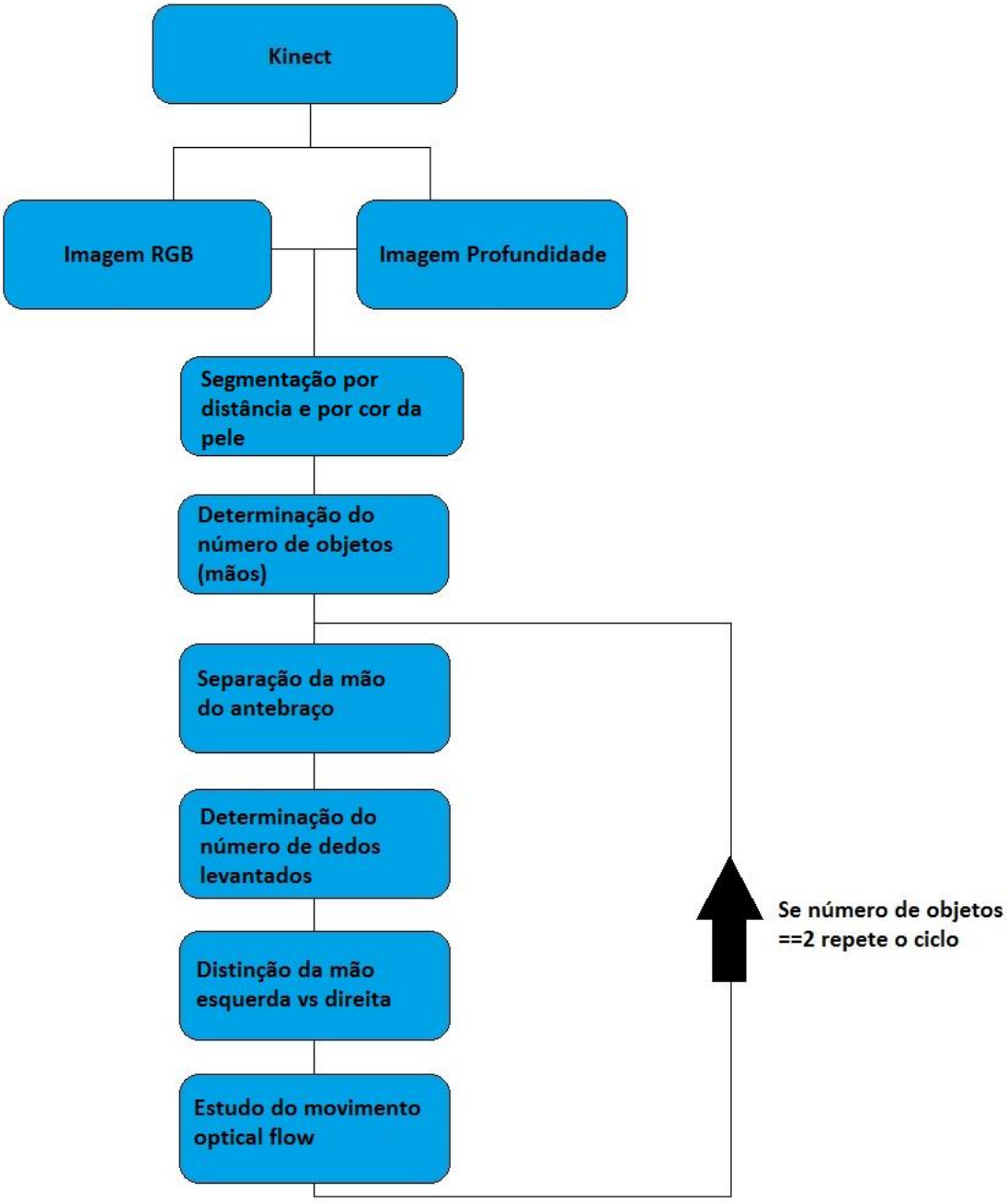


Figura 46. Flowchart para o processamento das mãos.

3.3 Deteção de dedos

Depois da segmentação das duas mãos o próximo passo é a determinação da localização das pontas dos dedos, de forma a ser possível saber quantos dedos estão levantados. Esta parte do trabalho é importante pois permite detetar com maior facilidade que tipo de gesto é que o utilizador está a fazer, enviando assim um comando gestual para o programa.

Apesar de já existirem alguns processos para determinar a posição das pontas dos dedos, como o método usado pelo Rui Barbosa [22] e o método de *Earth mover's distance* [25], o processo escolhido utiliza alguns métodos para melhorar a rapidez do processamento, facilitando assim a deteção das pontas dos dedos.

Baseado no trabalho "*Hand tracking and gesture recognition system for human-computer interaction using low-cost hardware*" [24], a determinação das pontas dos dedos pode ser feito através do ângulo de curvatura das pontas dos dedos.

Devido ao formato da mão, quando esta está aberta, a aplicação do algoritmo *Convex Hull* permite obter um conjunto de pontos que perfazem um invólucro convexo que contém todos os pontos da mão. O algoritmo obtém o involucro convexo mais pequeno possível. Esta é uma informação importante visto que a aplicação deste algoritmo para uma mão aberta difere para os casos em que um ou mais dedos não perfazem os pontos limites do invólucro convexo.

A figura abaixo mostra a aplicação do algoritmo de *Convex Hull* a uma região fronteira (linha verde). Os pontos vermelhos são o resultado obtido pelo algoritmo, a linha cinzenta é a união dos pontos limites que fazem o invólucro.

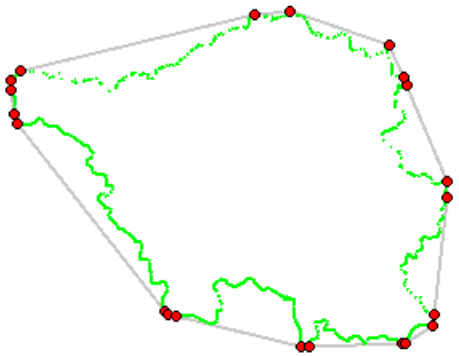


Figura 47. Aplicação da Convex Hull a uma imagem. Adaptado de [27].

Aplicando o algoritmo nas imagens da mão obtidas temos:

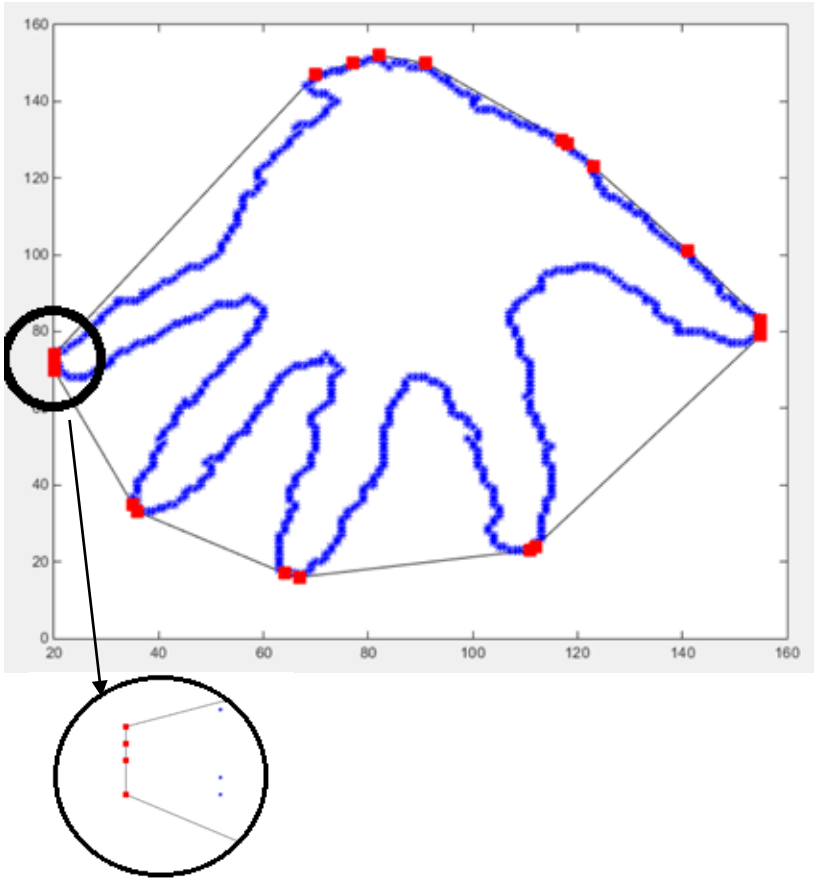


Figura 48. Determinação de todos os pontos que fazem a Convex Hull da fronteira da mão.

A figura 48 representa um gráfico da fronteira de uma mão onde foi aplicado o algoritmo de *Convex Hull*. As coordenadas estão invertidas em relação à imagem real para simplificação do estudo da aplicação do algoritmo na obtenção das coordenadas dos pontos.

Na imagem acima temos a fronteira da mão a pontos azuis e a pontos vermelhos os pontos que são obtidos pelo algoritmo de *Convex Hull* para a criação do invólucro. Esse invólucro é representado pela linha preta.

A partir da imagem repara-se que a ponta de todos os dedos pertence aos pontos obtidos pelo *Convex Hull*, o que é exatamente o que se pretende. No entanto, existem outros pontos que apesar de fazerem parte do invólucro não são pontos que pertençam aos dedos. Outra conclusão que se pode tirar, é que existem vários pontos muito próximos uns dos outros. Fazendo um *Zoom* à imagem repara-se que só na ponta do dedo mindinho existem 4 pontos.

Para fazer uma contagem do número de dedos correta, é necessário que por cada dedo haja apenas um ponto. Por esse motivo aplicou-se o algoritmo de redução da densidade de pontos numa região.

Para eliminar os pontos muito próximos basta introduzir um valor de distância cujos todos os pontos que se encontrem dentro desse valor sejam eliminados. A distância mínima

permitida para os exemplos referidos nas figuras abaixo foram de 8 pixéis, ou seja, todos os pontos que estivessem numa vizinhança de 8 pontos para o anterior eram eliminados.

Após a aplicação deste algoritmo, o resultado para a mesma mão da figura mostrada acima foi o seguinte:

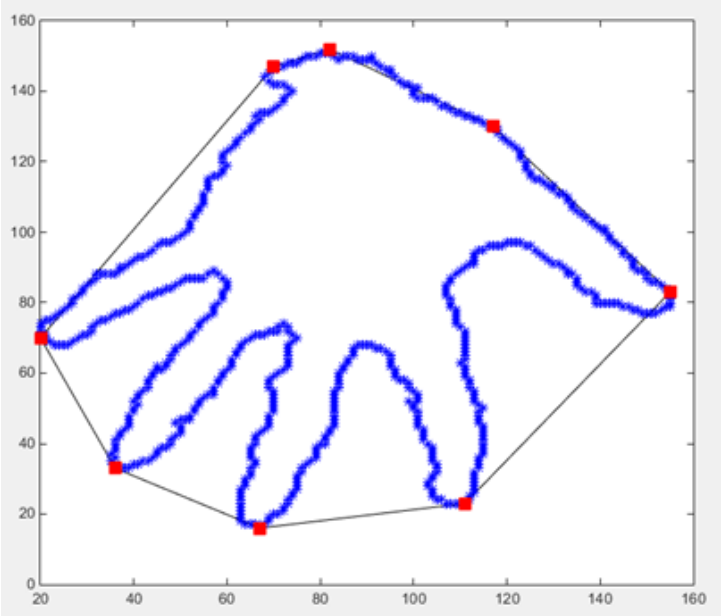


Figura 49. Aplicação do algoritmo de redução de densidade para eliminação dos pontos da Convex Hull muito próximos uns dos outros.

Agora, e como é pretendido, por cada dedo existe um único ponto, pontos esses que pertencem aos originais, mas cujos vizinhos foram eliminados por estarem muito próximos. Para além dos 5 pontos em cada dedo existem mais 3 pontos que são um resultado do *Convex Hull* mas que não pertencem a nenhum dedo e devem de ser eliminados.

A imagem da figura 49 mostra que o invólucro do *Convex Hull* corta regiões da mão, e na imagem anterior (figura 48) isso não acontecia. A razão pela qual isso acontece é que o desenho do invólucro não foi pedido na aplicação do *Convex Hull* na deteção de pontos da fronteira, mas sim aplicado depois de ter usado o algoritmo de redução de pontos, Ou seja, apesar dos pontos vermelhos observados na figura acima serem pontos da *Convex Hull* da fronteira da mão, o involucro é o resultado do *Convex Hull* dos pontos restantes do algoritmo de redução de pontos, e não da mão. Imagine-se que se apaga a fronteira e só estão presentes os pontos retirados do algoritmo de redução de pontos.

A aplicação do *Convex Hull* associada ao algoritmo de redução de pontos fez um trabalho fantástico na procura dos pontos pertencentes às pontas dos dedos. Contudo não deteta apenas as pontas dos dedos, também deteta regiões normalmente localizadas no pulso, regiões essas que causam falsos positivos. Para ignorar esses pontos usou-se o método *da K-Curvature* [24] [28], basicamente com todos os pontos resultantes da *Convex Hull* associado ao algoritmo de redução de pontos, vai-se estudar o ângulo da curvatura em que estão inseridos esses pontos.

Apesar de existirem alguns trabalhos com regras mais restritas para a classificação dos pontos, os resultados obtidos permitem a utilização do *K-Curvature* sem nenhuma condição adicional.

Para o estudo da *K-Curvature* vai ser necessário aceder a certas informações, como a posição que o ponto em causa está na matriz da fronteira da mão. Assim que a posição do ponto é encontrada vai ser preciso escolher um ponto que esteja numa posição Posição+K na matriz e outro ponto que esteja Posição–K. Com estes 3 pontos podemos criar dois vetores que partem da mesma origem, o ponto a ser estudado, e tem como destino os pontos vizinhos.

$$Vetor1 = [P(i), P(i+K)]$$
$$Vetor2 = [P(i), P(i-K)]$$

Equação 6. Vetores resultantes da K-Curvature

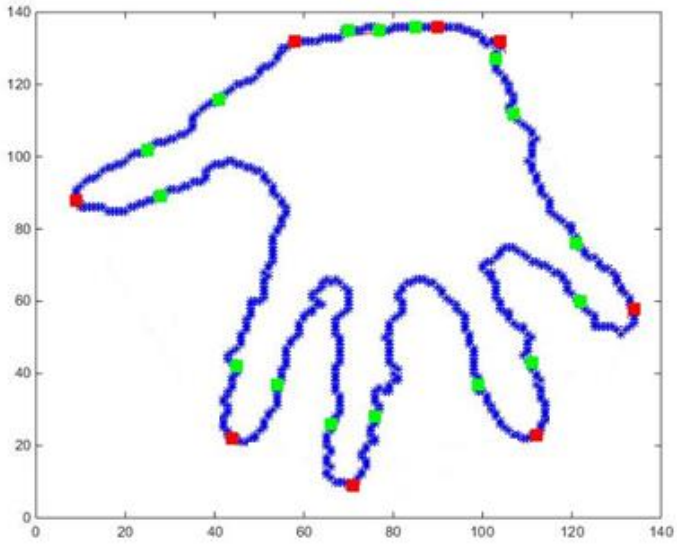


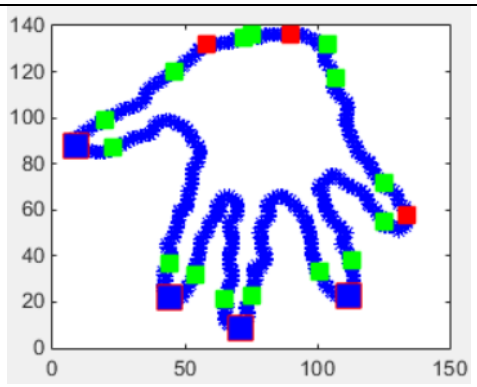
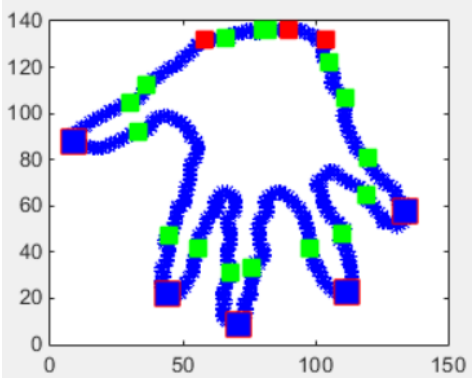
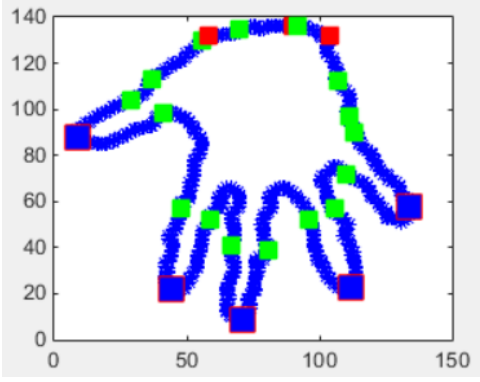
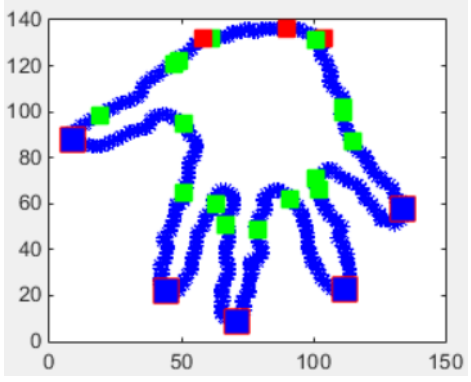
Figura 50. Representação dos pontos da K-Curvature (pontos a verde) para cada um dos pontos da Convex Hull + o algoritmo de redução de pontos (Pontos a vermelho).

Na imagem da figura 50, os pontos vermelhos correspondem aos pontos em estudo para determinação de dedos, e os pontos verdes correspondem aos pontos à frente e atrás a uma distância K na matriz da fronteira do ponto em estudo.

Os valores de K atribuídos são bastante importantes pois dependem do tamanho da mão. Se a mão estiver muito longe da câmara e se usar um K muito elevado os pontos resultantes podem ficar fora da fronteira do dedo.

Como a maior parte dos testes foram feitos a cerca de 1000mm da câmara o valor de K usado foi de 25. A escolha deste valor foi atribuída para que os pontos resultantes ficassem na fronteira do dedo respetivo. Os casos mais graves correspondem aos dedos mais pequenos, o polegar e o mindinho, sendo que valores muito acima de 25 podem ficar fora do dedo e valores muito abaixo podem-se tornar também problemáticos se os pontos ficarem mesmo na ponta dos dedos.

Tabela 2. Evolução da posição dos pontos à frente e atrás do ponto em estudo.

K	Imagem
15	
25	
35	
45	

A razão para que a posição dos pontos retirados através do *K-Curvature* seja na parede dos dedos, é que com os vetores que se obtêm, pode retirar-se os valores dos ângulos entre esses vetores.

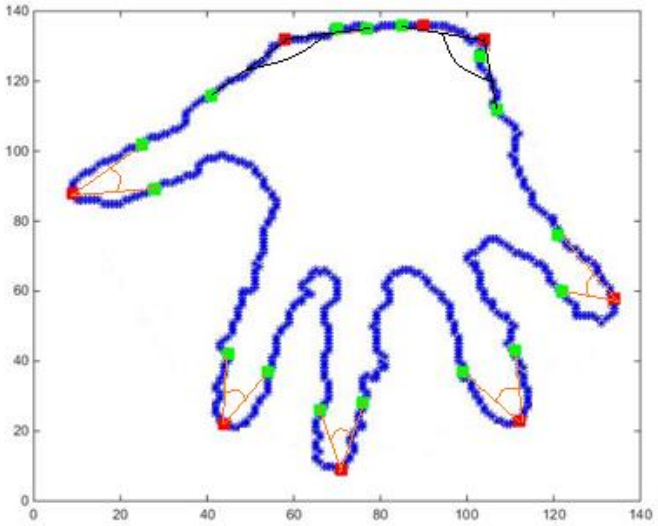


Figura 51. Representação dos vetores entre o ponto da Convex Hull + Reducem com os seus respectivos pontos da K-Curvature e a visualização do ângulo formado entre esses mesmos vetores.

Na figura 51 observa-se que os ângulos entre os vetores nas pontas dos dedos (representados a laranja) são muito menores que os ângulos dos vetores nos pontos do pulso (representados a preto). Com esta informação temos as ferramentas necessárias para separar os pontos das pontas dos dedos, dos pontos do pulso.

$$\theta = \cos^{-1}\frac{\vec{A}.\vec{B}}{|\vec{A}||\vec{B}|}$$

Equação 7. Equação para a determinação do ângulo entre dois vetores.

Aplicando a simples equação para a determinação de ângulos entre dois vetores [equação 2]: O para saber qual dos pontos pertence à ponta dos dedos, o ângulo entre os vetores resultantes tem de ser inferior a 60º. O valor 60º foi obtido a partir de alguns testes experimentais. A maior parte dos ângulos obtidos tinha um valor inferior a 60º, contudo alguns pontos pertencentes aos dedos tinham valores de ângulos entre vetores perto dos 70º. Os pontos pertencentes à região do pulso têm na sua maioria valores de ângulos superiores a 100º. Ocasionalmente aparecem valores com menos de 70º.

Caso o limite máximo de ângulo permitido seja 70º, ocasionalmente vão-se registar falsos positivos na região do pulso.

No entanto a maior causa de erros neste método vem da aplicação do algoritmo do *Convex Hull*. O *Convex Hull* vai procurar os pontos da fronteira que permitem a criação de um polígono convexo que envolva toda a região delimitada pela fronteira da mão. Se um dedo, como o indicador, tiver um tamanho muito pequeno (que pode acontecer por exemplo devido a

dedo amputado), dedo dobrado ou até mesmo por falha na aquisição do contorno da mão pela segmentação, o *Convex Hull* não vai considerar o ponto da ponta do dedo para a criação do polígono convexo.

Na imagem em baixo [ver figura 52] está representada uma silhueta da mão com o dedo indicador com um tamanho menor do que o normal.

Figura 52. Representação de uma mão com o dedo indicador com tamanho que não permite a criação de um ponto de Convex Hull.

Aplicando o algoritmo desenvolvido para a imagem da figura 52, obtemos o resultado demonstrado na figura 53.

(NOTA: os quadrados azuis com bordo vermelho indicam a presença de dedo, e as linhas pretas são o polígono resultante do *Convex Hull* representados pelos quadrados vermelhos).

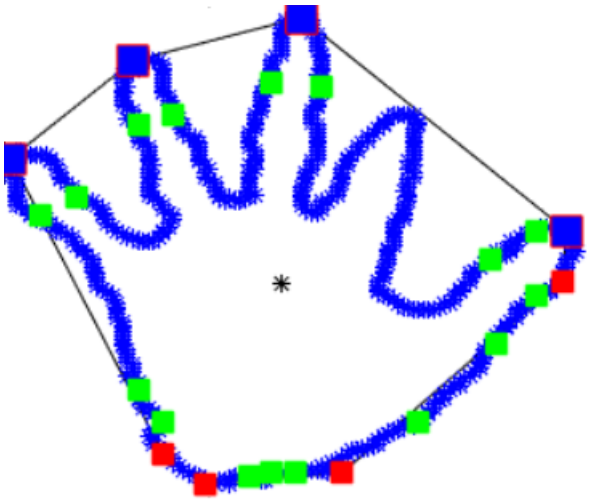


Figura 53. Falha em detetar o dedo indicador pela Convex Hull.

Como é possível observar na figura 53, a ponta do dedo indicador não é detetada como dedo nem sequer é assinalado como ponto de estudo.

Para evitar esses erros estudou-se uma alternativa. Recorrendo mais uma vez à utilização da *K-Curvature* mas aplicada a todos ou pelo menos a grande parte dos pontos que fazem a fronteira da mão. Este método iria detetar vários pontos por dedo, visto que os pontos

muito próximos entre si na ponta dos dedos têm uma curvatura muito parecida, problema esse que pode ser resolvido com a aplicação posterior do algoritmo para redução de pontos. No entanto aplicação deste método iria também detetar as regiões côncavas entre os dedos.

Aplicando o método proposto obteve-se o seguinte resultado:

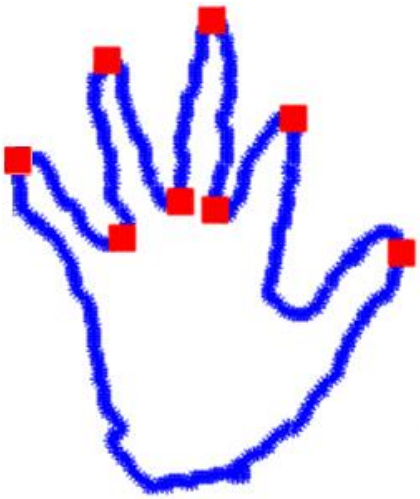


Figura 54. Resultado da aplicação da K-Curvature a todos os pontos da fronteira da mão com o dedo indicador com tamanho inferior.

Os pontos vermelhos representam todos os pontos com requisitos requeridos na condição *K-Curvature*. No entanto, e como é observável e já espectável, nem todos os pontos pertencem às pontas dos dedos. Para separar os pontos que são depressões (estão entre dedos), dos pontos que pertencem às pontas dos dedos será necessário criar mais condições.

Uma das soluções seria a eliminação por distância, ou seja, todos os pontos (vermelhos) que estivessem a uma distância inferior a um valor determinado do centro da palma da mão já calculado anteriormente, seriam ignorados. A maior desvantagem deste método reside no facto de ser necessário estudar a *K-Curvature* de muitos pontos, sendo que esse estudo requer uma capacidade computacional superior à da utilização da *Convex Hull* que elimina *A priori* bastantes pontos.

Em termos de tempo de processamento, para a imagem da figura 54, o método usando o *Convex Hull* necessitou apenas de 0.12 segundos, enquanto o estudo a todos os pontos da fronteira necessitou de 0.24 segundos, ou seja o dobro do tempo.

Contudo, a possibilidade de uma exatidão no número de dedos, torna-se bastante aliciante. O primeiro método só reconhece 4 dedos e o segundo já reconhece os 5. Após vários testes verificou-se que o método com recurso a *Convex Hull*, apesar de ser mais falível, os erros na determinação dos dedos não são muito mais elevados.

Algoritmo para determinação de dedos:

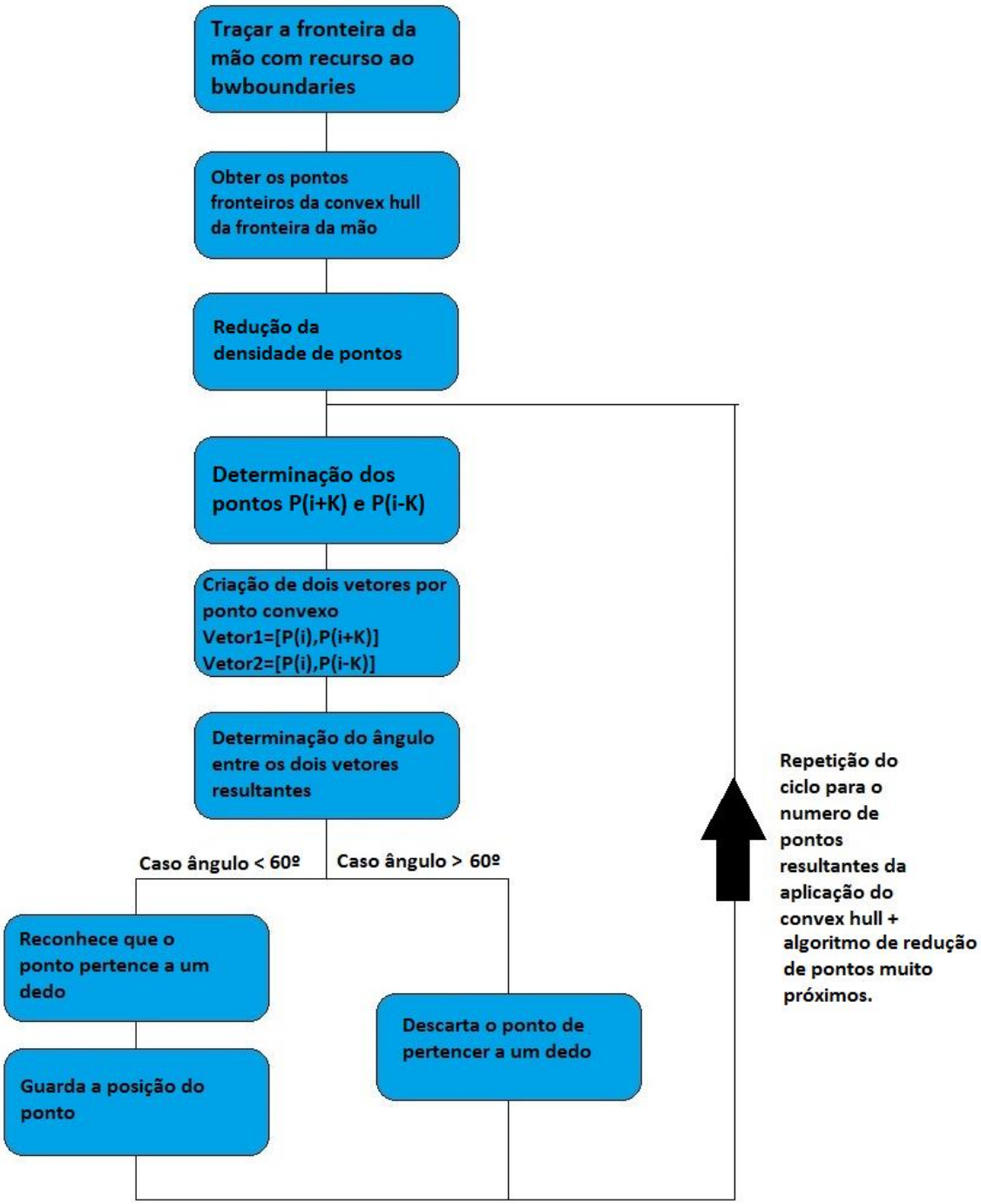


Figura 55. Flowchart para determinação dos dedos.

3.4 Distinção entre mão esquerda e direita.

O próximo passo a realizar é a distinção da mão direita e da mão esquerda. No que toca a classificação de gestos por movimento, a distinção pode ser bastante útil, por exemplo, em termos de rotação. Se a mão direita estiver a subir e a esquerda a descer pode criar um comando para uma rotação anti-horário, enquanto uma mão direita a descer e uma mão esquerda a subir cria uma rotação horária.

Intuitivamente sabe-se que numa imagem com duas mãos em estado de repouso, ou seja, ambas as mãos apontadas diretamente para a câmara sem cruzamento de braços, a mão direita aparecerá mais à esquerda na imagem enquanto a mão esquerda aparecerá mais à direita na imagem.

O problema é que isso funciona muito bem caso as mãos estejam estáticas e apareçam ambas na imagem. Se o utilizador por algum motivo cruzar os braços, a informação aparecerá errada, pois a mão esquerda aparecerá mais próximo do lado esquerdo, e a outra mão vice-versa. Caso apareça apenas uma mão torna-se impossível saber qual das mãos está levantada.

Quando apenas uma mão está virada para a câmara, saber se é a mão direita ou a esquerda pode ter as suas vantagens. Com esta motivação começou-se por identificar diferenças entre as mãos, que as permitissem distinguir.

Em termos de parecenças físicas não há nada que permita fazer uma distinção. Nenhuma mão é maior que a outra, nenhuma mão tem mais dedos que a outra. Contudo uma mão é simétrica da outra, e o ponto que mais evidencia a simetria das mãos é o polegar.

A mão esquerda tem o polegar a apontar para o lado esquerdo enquanto a mão direita tem um polegar a apontar para o lado direito. Sabendo então qual dos dedos é o polegar, será possível com relativa facilidade fazer a distinção entre as mãos.

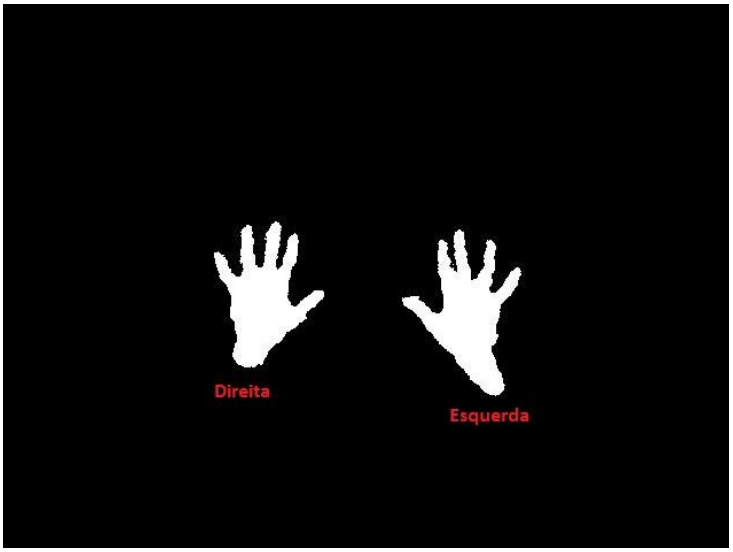


Figura 56. Objetivo a alcançar. Distinção entre a mão esquerda da direita.

A deteção das coordenadas das pontas dos dedos não faz a distinção entre dedos, apenas dá a informação das coordenadas de um ponto que pertence à ponta dos dedos.

Analisando a imagem da mão, o dedo polegar é distinguível a um humano, que com bastante facilidade pode dizer qual dos dedos é o polegar. Uma das características do dedo polegar é que ele ou é o primeiro dedo, ou o último (a contar da esquerda para a direita). Infelizmente caso a mão esteja rodada a distinção por posição relativa torna-se complicado, ainda que, se o dedo polegar não estiver apresentável pode-se contar um outro dedo erradamente.

Na imagem abaixo temos uma situação em que se torna muito complicado fazer a distinção:

Figura 57. Exemplificação de uma das várias posições que as mãos podem ter. A mão de cima neste caso não é distinguível.

Um dos desafios é criar um código que permita a distinção mesmo que a mão esteja rodada como observado na imagem da figura 57. A mão de cima na imagem tem o dedo polegar oculto. Como a distinção será feita a partir do dedo polegar, caso esse dedo esteja oculto ou não seja possível identificá-lo, não vai ser possível saber que mão é que está presente.

Para encontrar o dedo polegar algumas ideias foram consideradas:

Comprimento do dedo:

O dedo polegar é dos mais pequenos em termos de comprimento, apenas o mindinho pode ter um tamanho parecido, esta é uma informação bastante importante que será usada mais adiante.

Largura do dedo:

O dedo polegar é também o dedo cuja largura é a maior de todos os dedos. Contudo com a resolução muito baixa da imagem em profundidade as larguras dos dedos tem bastantes discrepâncias de frame para frame, por vezes na imagem binária resultante da profundidade o dedo polegar não apresenta a maior largura.

A partir da pesquisa bibliográfica [24] encontrou-se uma maneira fiável para encontrar o dedo polegar a partir de uma imagem da mão, fazendo uso de algoritmos úteis no passado como o caso do *Convex Hull*. É possível criar um polígono que envolva toda a mão. Como foi referido anteriormente a criação do polígono é feita depois de aplicado o algoritmo de redução de pontos. Assim pode-se garantir que há sempre uma separação entre áreas, entre os dedos [ver figura 58].

De seguida basta subtrair a figura da mão à do polígono.

Figura 58. Representação das regiões entre o polígono convexo e a mão.

Estudando a área entre os dedos repara-se que a região entre o polegar e o indicador é a maior.

Com a região entre os dedos polegar e indicador será possível com relativa facilidade determinar a localização do dedo polegar.

Figura 59. Seleção da maior área.

A região vermelha é o objeto com a maior área.

O estudo da região com a maior área é um bom princípio para determinar a localização do polegar mas tem os seus inconvenientes. Como o objetivo é o processamento de uma mão em constante mudança (vários gestos) e movimento, há a possibilidade de a maior área não corresponder à região entre o dedo polegar e indicador. Essa situação pode causar numa distinção errada, pode-se ter um polegar indicado numa região errada.

Exemplo para a imagem seguinte:

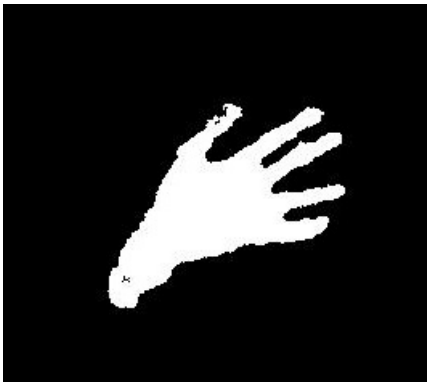


Figura 60. Representação da mão.

Aplicando o mesmo método referido anteriormente:

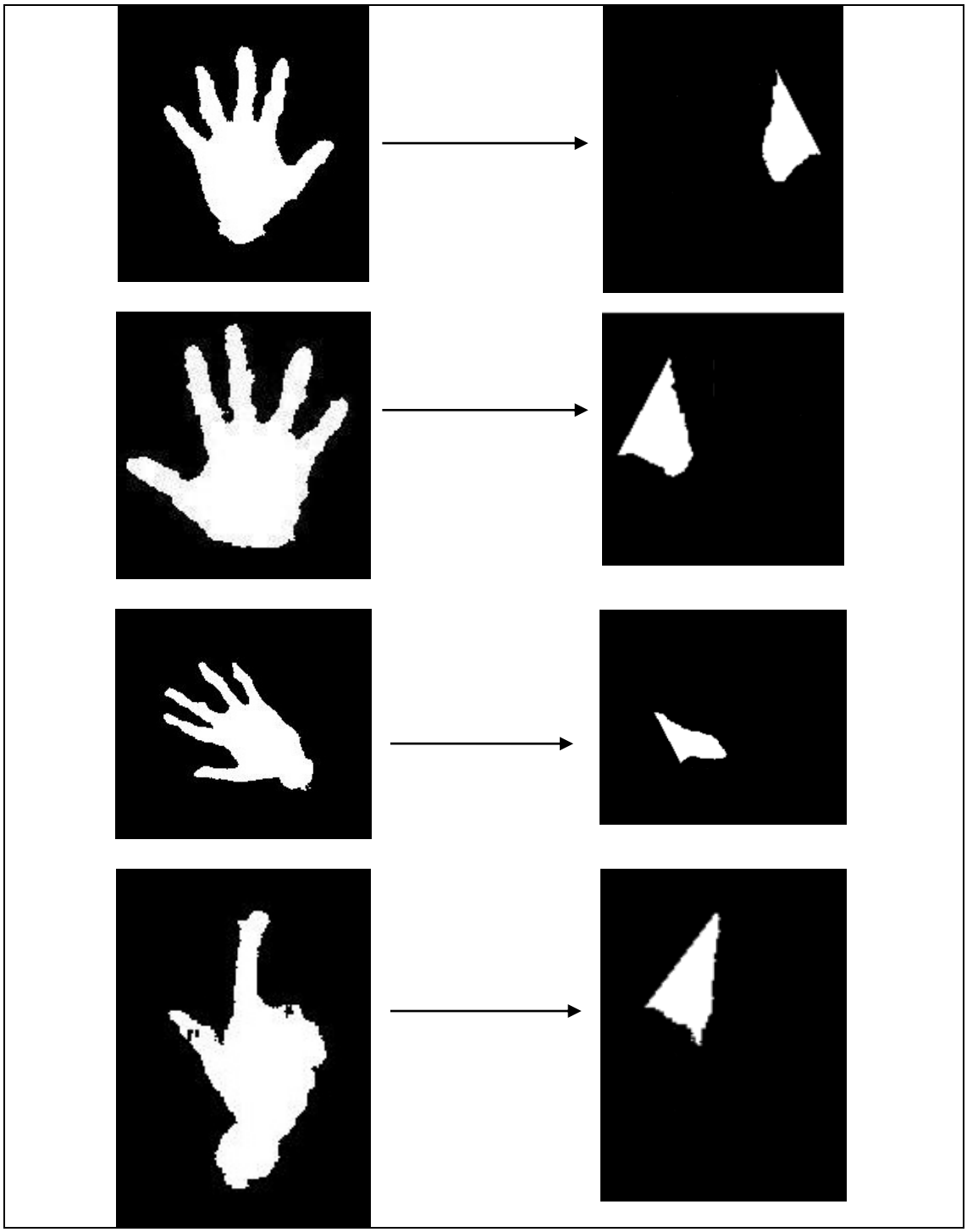
Figura 61. Representação da região com a maio área.

A região com a maior área (região representada a vermelho) não é desejável pois não se encontra entre o dedo polegar e o indicador, tornando-se assim impossível detetar o dedo polegar e o dedo indicador.

Para resolver esta questão é necessário obter mais informações das regiões. Obtendo várias imagens para estudo, reparou-se que a região entre o dedo polegar e o indicador tem uma forma distinta de todas as outras regiões.

Em baixo na coluna esquerda temos uma representação de uma silhueta da mão, enquanto na esquerda obtemos a região respetiva de cada imagem da região entre o dedo polegar e o dedo indicador.

Tabela 3. Associação das maiores regiões entre os dedos para as diversas formas da mão.



As regiões entre o dedo polegar e o indicador das diferentes silhuetas das mãos têm parecenças. Com esta informação podemos encontrar uma maneira de diferenciar esta região de todas as outras.

Por isso decidiu-se utilizar o fator de forma para obter a região entre o dedo polegar e o dedo indicador.

$$ff = \frac{4 * \pi . Area}{Perimetro^2}$$

Equação 8. Equação do fator de forma.

Após a determinação do fator de forma [ver equação 8] das regiões entre os dedos pretendidos de várias imagens chegou-se a um valor mínimo e máximo admissível.

$$0.4 < ff < 0.65$$

Equação 9. Limites do valor de fator de forma para a região entre os dedos polegar e indicador.

Logo a região pretendida tem de ter um valor de fator de forma compreendido entre os valores acima indicados.

Para evitar fazer o cálculo do fator de forma de todas as regiões, e visto que na maior parte dos casos (situação observada experimentalmente) a região com a maior área é de facto a região entre o dedo polegar e o dedo indicador, optou-se por deixar a escolha da região com a maior área associada ao estudo do fator de forma.

O programa determina qual das regiões tem a maior área e de seguida calcula o seu fator de forma. Caso o fator de forma não esteja dentro dos limites anteriormente definidos, o programa entra num ciclo em que procura a segunda maior área e volta a calcular o seu fator de forma, até encontrar uma região com o fator de forma dentro dos valores aceitáveis. Caso nenhuma das regiões seja aceite no fim do ciclo, a mão não é identificável.

Assim evita-se o cálculo de todos os fatores de forma de todas as regiões. Apesar de em termos de processamento não ser muito pesado, é evitável.

Agora que obtivemos a região pretendida é necessário descobrir onde fica o dedo polegar e o dedo indicador. A maior parte das vezes a região assemelha-se a um triângulo, em que dois dos seus vértices se encontram perto das pontas dos dedos. Embora não correspondam exatamente ao ponto que foi calculado anteriormente para a determinação das pontas dos dedos, está muito próximo.

Na figura 62 é possível observar os 3 pontos vértices do triângulo:

Figura 62. Vértices da região entre o dedo polegar e o indicador.

Para a determinação dos vértices do triângulo, foi usado o algoritmo para obtenção dos pontos extremos de uma região.

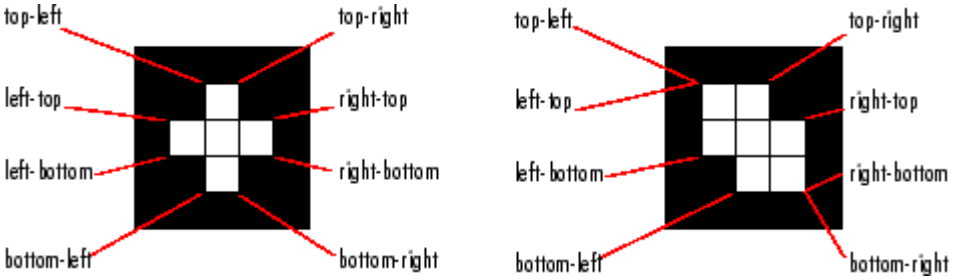


Figura 63. Pixéis extremos de uma região. Créditos MATLAB

Sendo o formato da região em causa muito parecido com um triângulo, a aplicação do algoritmo para os pontos extremos oferece diversos pontos que se situam perto dos dedos e da zona côncava ente os dedos (a zona mais baixa da depressão).

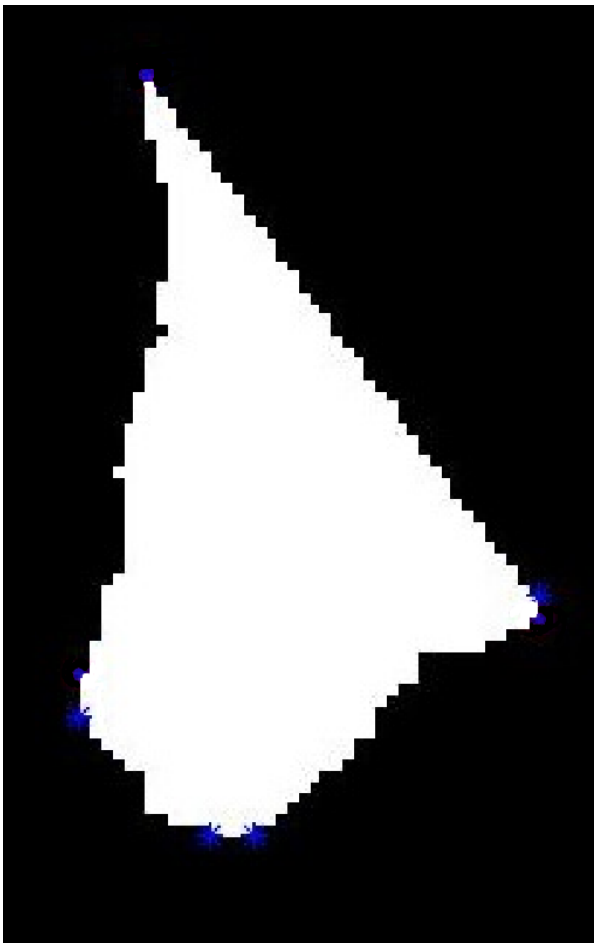


Figura 64. Localização dos pontos extremos da região.

Após a aplicação do algoritmo para os pontos extremos obtém-se muitos pontos juntos, que podem ser reduzidos aplicando o algoritmo usado anteriormente para redução da densidade.

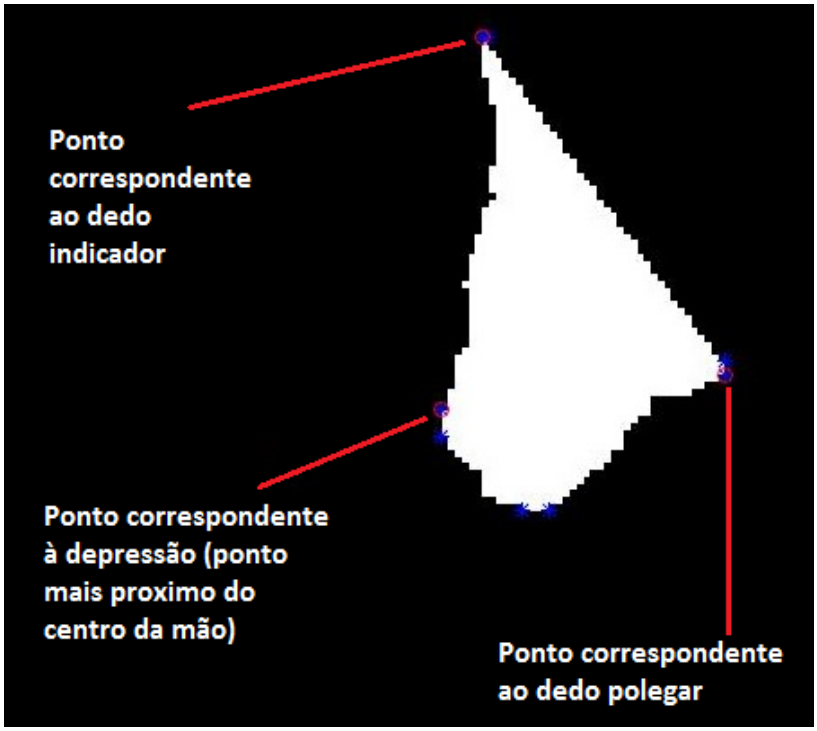


Figura 65. Localização dos pontos resultantes após a aplicação do algoritmo de redução de pontos próximos (Pontos com circulo vermelho).

Na imagem da figura 65 temos apenas os pontos mais interessantes, em que dois deles estão muito próximos dos pontos das pontas do polegar e indicador, e um terceiro ponto junto à zona pertencente à depressão da região. É importante salientar que depois de aplicado o algoritmo dos pontos extremos, mais o algoritmo de redução de pontos próximos é necessário obter 3 pontos.

O formato da região permite a obtenção desses 3 pontos com alguma facilidade, mas nem sempre é possível.

Após a obtenção desses 3 pontos é preciso saber qual corresponde ao dedo indicador, dedo polegar e o mais próximo ao centro da mão.

Determinam-se as distâncias dos 3 pontos ao ponto correspondente ao centro da palma da mão calculado no capítulo da segmentação da mão. O ponto que tiver a menor distância será então o ponto da zona da depressão, como é observado na imagem da figura 66, a linha verde corresponde à mínima distância. Assim, um dos três pontos já está decifrado, e será designado por ponto da depressão.

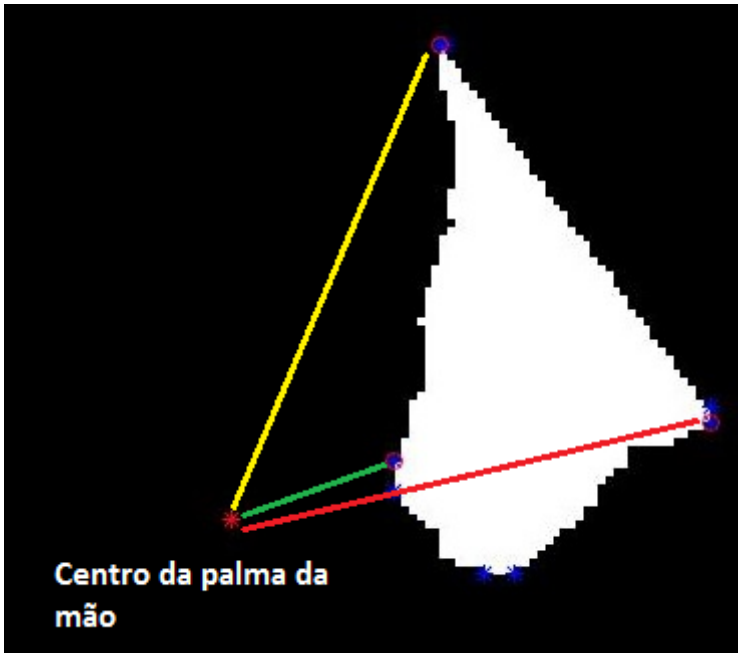


Figura 66. Representação dos vetores entre os pontos extremos e o ponto central da palma da mão.

A partir do ponto da depressão volta-se a calcular a distância euclidiana aos outros restantes pontos.

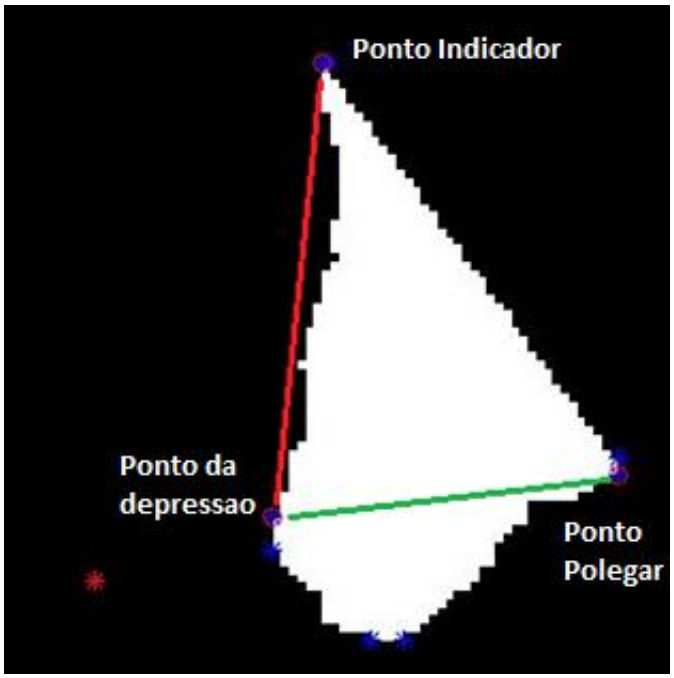


Figura 67. Vetores entre os vértices correspondentes às pontas dos dedos e o vértice mais próximo do centro da palma da mão.

A menor distância corresponde ao vetor que liga o ponto da depressão com o ponto do polegar (vetor polegar representado a verde). Por eliminação o outro vetor liga o ponto da depressão com o ponto indicador (vetor indicador representado a vermelho). Apesar do ponto polegar não corresponder exatamente às coordenadas da ponta do polegar (o mesmo

acontece para o ponto indicador com a ponta do indicador), devido à subtração do polígono com a imagem da mão não ser feita a partir dos pontos considerados dedos [ver figura 58] a aproximação é aceitável permitindo assim obter as condições que perfazem a distinção entre as duas mãos.

Com os dois vetores obtidos aplica-se o cálculo do produto externo entre os mesmos.

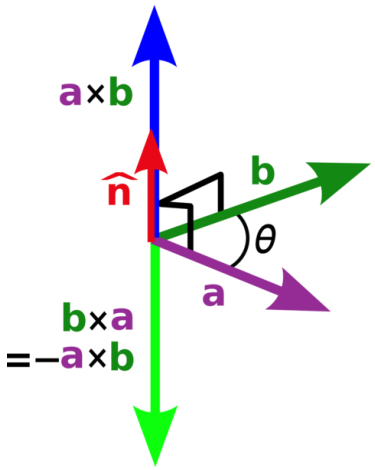


Figura 68. Representação do produto externo.

Figura 69. Representação dos vetores em duas mãos.

Na representação da imagem acima, aplicando o produto externo do menor vetor (vetor polegar) com o vetor indicador, é possível obter um terceiro vetor cuja terceira dimensão será para fora ou para dentro da imagem segundo a regra da mão direita.

Para que seja consistente, a aplicação do produto externo terá sempre de ser do vetor de menor dimensão para o vetor de maior dimensão.

Segundo o sistema de eixos da imagem para a mão esquerda, o produto externo do vetor verde com o vetor vermelho será Z:

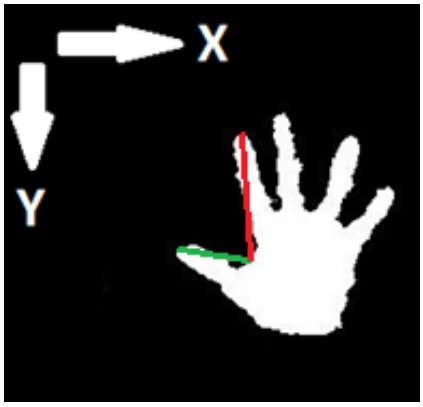


Figura 70. Representação do referencial da imagem. E apresentação dos vetores para a mão esquerda.

Enquanto que para a mão direita será –Z:

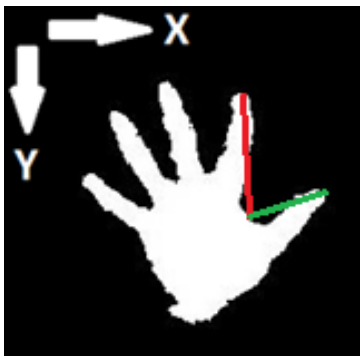


Figura 71. Representação do referencial da imagem. E apresentação dos vetores para a mão direita.

Com este método é possível com alguma confiança fazer a distinção das mãos esquerda e direita. A sua fiabilidade no entanto não é 100%. Como referido acima, o requisito para que a distinção seja feita é que o dedo polegar e o dedo indicador estejam levantados.

A figura abaixo representa a condição mínima para a distinção.

Figura 72. Para fazer a distinção é preciso que o dedo polegar e o dedo indicador estejam levantados.

A desvantagem do método reside no facto de que a obtenção dos pontos extremos pode não ser a desejável.

Para os pontos próximos das pontas dos dedos, ou seja, o ponto indicador e o ponto polegar, são determinados com bastante fiabilidade a partir do algoritmo para os pontos extremos. O problema situa-se no terceiro ponto, aquele que se situa mais próximo do centro da palma da mão. A zona onde o ponto é determinado não é tão pontiaguda como acontece com os outros dois pontos. A maior parte das vezes até se situa numa curva, resultando assim numa obtenção do ponto em qualquer parte da curva.

Como a obtenção do ponto pode ser muito dispersa nessa região da curva, o ponto pode ser determinado numa localização cuja sua distância ao ponto indicador seja menor que a distância ao ponto polegar. Caso isso aconteça a determinação do vetor polegar será feita para o dedo indicador e o vetor indicador será feito para o dedo polegar. Com os vetores trocados a distinção entre a mão esquerda e mão direita podem estar trocadas. Para evitar essa situação decidiu-se eliminar o terceiro ponto e substitui-lo pelo centro da palma da mão. A utilização do centroide, como já foi referido anteriormente, pode encontrar-se em vários locais dependendo da forma da mão, o que pode causar erros na distinção. Para obter um resultado mais consistente, a utilização do centro da palma da mão é a escolha mais acertada. Note-se também que o centro da palma da mão está sempre mais perto da ponta do dedo polegar do que da ponta do dedo indicador, minimizando assim as hipóteses de obter vetores trocados para a aplicação do produto externo.

Figura 73. Alguns gestos que permitem a distinção entre mão esquerda e direita.

Figura 74. Alguns gestos que não permitem a distinção entre mão esquerda e direita.

Juntando os algoritmos realizados até agora e aplicando-os ao número de objetos presentes numa imagem segmentada num fluxo de vídeo continuo, é possível em tempo real fazer a distinção das duas mãos e identificar as coordenadas das pontas dos dedos.

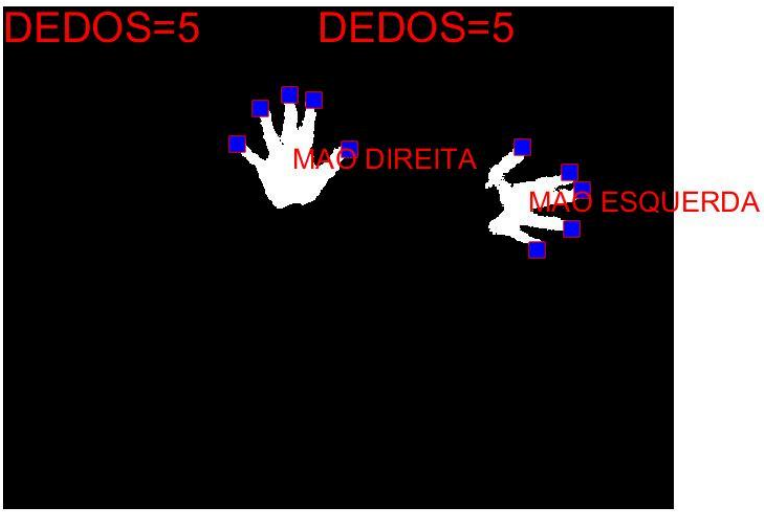


Figura 75. Frame processado de um fluxo de vídeo em profundidade adquirido da Kinect.

3.5 Utilização do *Optical Flow* para gestos em movimento.

O trabalho a ser desenvolvido tem como objetivo o controlo de programas/aplicações informáticas, portanto o movimento é um aspeto importante a ser analisado. O movimento do rato é imprescindível no computador pessoal. No caso dos *tablets* a indicação de gestos em movimento não é feito com a utilização do rato mas sim com os dedos do utilizador. Instruir comandos ao computador/*tablet* como por exemplo, arrastar objetos, apontar para determinados locais, *Zoom in/Zoom out*, são todos feitos com recurso a movimento.

Figura 76. Exemplificação do funcionamento do Zoom em Tablets. Créditos: Sony

Com as mãos o movimento é fundamental, e com isso é preciso traduzir o movimento em comandos/instruções. Ao contrário do rato as mãos podem estar sempre em movimento perante uma câmara. Alguns desses movimentos serão instruções enquanto outros serão movimento no vazio, sem o utilizador querer transmitir nenhuma informação para ser processada. Por esses motivos o movimento da mão, ou das mãos, terá que ser classificado, para ser ou não interpretado como comandos para atuação.

A utilização do *Optical Flow* permite saber com bastante facilidade não só a direção do movimento como a velocidade da mão, obtidas a partir do vetor resultante da posição final e inicial.

Para o controlo do cursor do rato para além da utilização do *Optical Flow*, foi considerado o uso da localização por coordenadas absolutas.

A utilização do *Optical Flow* permite o controlo do rato através da soma de vetores a posições conhecidas.

O método das coordenadas absolutas aproxima-se mais com o sistema usado em *Tablets*, em que a posição do cursor é sempre igual à do dedo. Se levantar o dedo do ecrã e o colocar num ponto qualquer do ecrã o rato vai ao encontro do dedo.

Para este trabalho optou-se pela utilização do sistema *Optical Flow*.

Existem vários algoritmos para o cálculo do *Optical Flow* [29], sendo que os usados para teste foram:

- Horn-Schunck
- Lucas-Kanade

O objetivo de ambos os algoritmos é resolver a seguinte equação adaptada de [30] [31] [32]:

$$I_x u + I_y v + I_t = 0$$

Equação 10. Equação de restrição do Optical Flow.

Em que I_x, I_y e I_t são as derivadas espaço-temporal do brilho da imagem

u é o *Optical Flow* horizontal

v é o *Optical Flow* vertical

Lucas-Kanade: Divide a imagem em secções pequenas e assume uma velocidade constante para cada região. O método também considera que o objeto em causa não se move muito de frame para frame [30].

Horn-Schunck: Assume que o *Optical Flow* é suave para a imagem toda. E tenta minimizar o erro da seguinte equação [30] [32]:

$$E = \int\int (I_x u + I_y v + I_t)^2 dxdy + \alpha \int\int \left\{ \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 \right\} dxdy$$

Equação 11. Equação de minimização de erro.

α Corresponde ao valor de suavidade desejado.

$$u_{x,y}^{k+1} = \bar{u}_{x,y}^{k} - \frac{I_x \left[I_x \bar{u}_{x,y}^{k} + I_y \bar{v}_{x,y}^{k} + I_t \right]}{\alpha^2 + I_x^2 + I_y^2}$$

$$v_{x,y}^{k+1} = \bar{v}_{x,y}^{k} - \frac{I_x \left[I_x \bar{u}_{x,y}^{k} + I_y \bar{v}_{x,y}^{k} + I_t \right]}{\alpha^2 + I_x^2 + I_y^2}$$

Equação 12. Campo de velocidades.

Ambos foram experimentados durante a realização da dissertação.

Os resultados em termos de tempo de processamento não variam muito. Apesar do algoritmo Horn-Schunck ter como desvantagem ser mais sensível ao ruido, é o que se comporta melhor quando o *Optical Flow* é aplicado para o controlo do cursor do rato.

O algoritmo de *Optical Flow*, assume que o brilho de um determinado pixel não se altera de frame para frame, sendo assim possível identificar a posição do pixel no frame inicial e final.

A quantificação do movimento é dado pelo algoritmo em forma de números complexos em que a componente vertical corresponde ao valor real enquanto a componente horizontal corresponde ao valor imaginário.

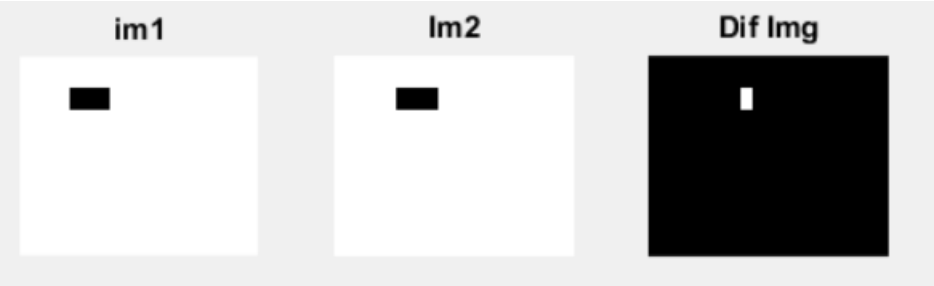


Figura 77. Movimento de um objeto binário. Translação feita na horizontal e da esquerda para a direita.

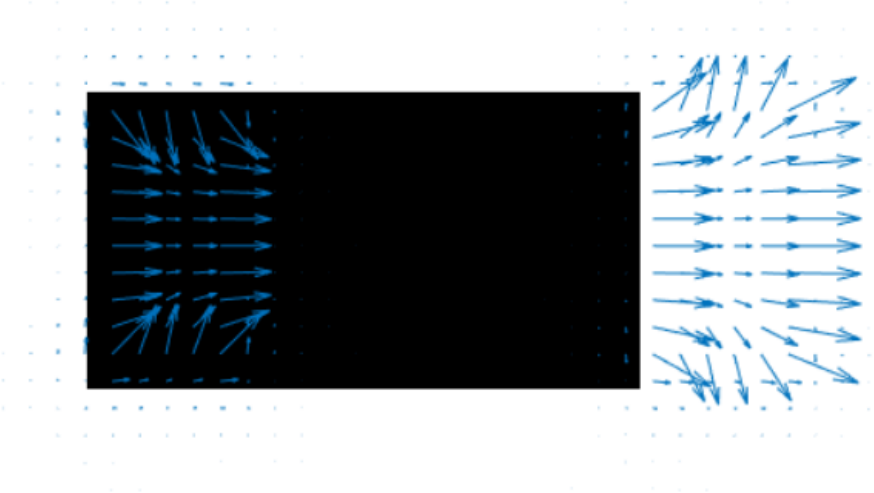


Figura 78. Vetores resultantes do movimento descrito na figura 77.

Na imagem da figura 78 [ver também figura 79] estão representados os vetores do movimento descrito. Como o movimento realizado foi na horizontal e da esquerda para a direita, seria lógico que os vetores representados aparecessem todos a apontar para a esquerda e na horizontal, mas tal não sucede. Apesar de todos os vetores apontarem para a esquerda nem todos apontam na horizontal.

Essa situação torna-se mais evidente quando se utiliza o algoritmo para o movimento da mão.

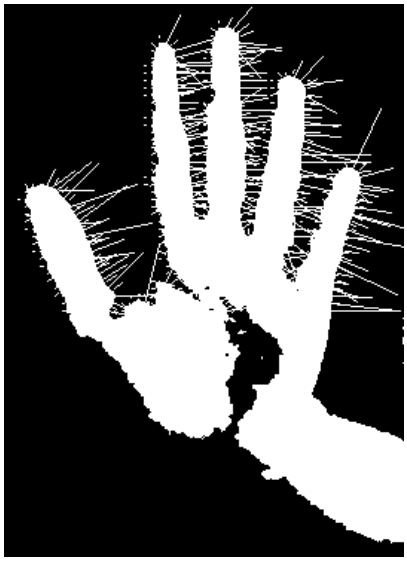


Figura 79. Vetores resultantes após um movimento na horizontal da esquerda para a direita.

A imagem da figura 79 representa um movimento realizado com a mão e a obtenção dos vetores de movimento, observando-se que existem alguns vetores com componentes verticais.

Estes vetores podem ser resultado do problema da abertura do diafragma, em que a direção do movimento do objeto não corresponde ao *Optical Flow* determinado [ver figura 81].

O problema da abertura do diafragma acontece devido à pequena abertura (janela de observação) com que o algoritmo trabalha. Como o algoritmo analisa regiões pequenas em vez do objeto num todo, é difícil conseguir saber a direção do movimento numa área muito pequena, como representado na figura 82.

Nos três casos a tira está a mover-se em direções diferentes, apenas a direção das listas é igual, contudo, caso se esteja só a observar o movimento das listas dentro do círculo (janela de observação), distinguir o movimento das tiras é impossível.

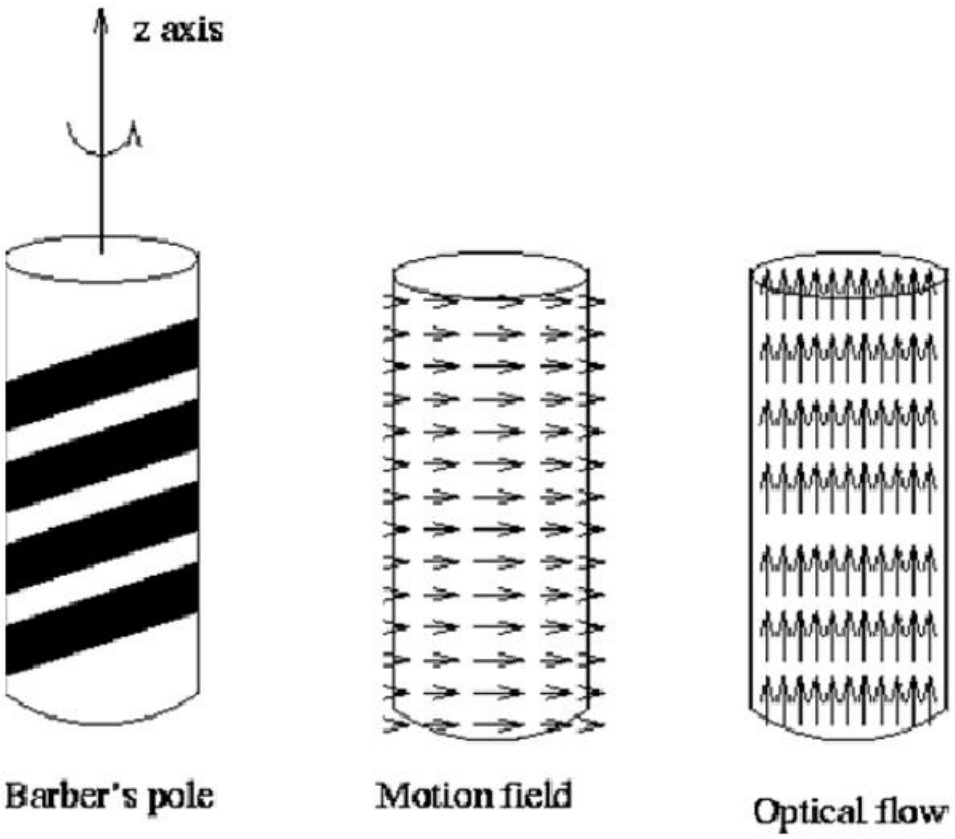


Figura 80. Representação do problema da apertura no Optical Flow.

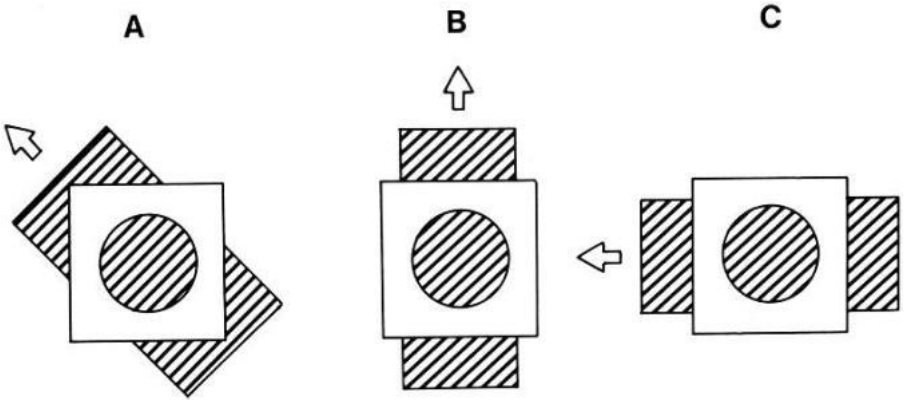


Figura 81. Representação de uma janela de visualização.

Existem algumas condições que podem minimizar o erro da abertura do diafragma, como assunções de suavizações locais do *Optical Flow* como por exemplo assumir que os pixéis de uma certa região vizinha se deslocam com velocidade ou aceleração constantes [33]. O algoritmo fornecido tem a opção de controlar a suavidade [equação 11] pretendido, em que se o movimento entre dois frames consecutivos for muito baixo utiliza-se um valor positivo pequeno, valores de movimento elevados entre dois frames usa-se um valor positivo maior.

Recorrendo a métodos experimentais os valores para os quais os resultados foram mais aceitáveis estavam compreendidos no intervalo [1,5].

Depois do sistema *Optical Flow* aplicado é necessário classificar os movimentos obtidos. Para isso, e como se obtém um grande número de vetores de movimento para a mão, é calculada a média de todos os vetores, quer para a componente vertical quer para a horizontal.

A partir do *Optical Flow*, só é possível determinar movimentos feitos no plano (X,Y), ou seja, movimentos 2D, sendo que movimentos no eixo ótico, Z, não foram determinados na realização deste trabalho.

A partir dos valores médios das componentes verticais e horizontais do movimento registado, é possível classificar o movimento num gesto para atuação.

Alguns dos gestos previstos para atuação estão ilustrados na figura 82.

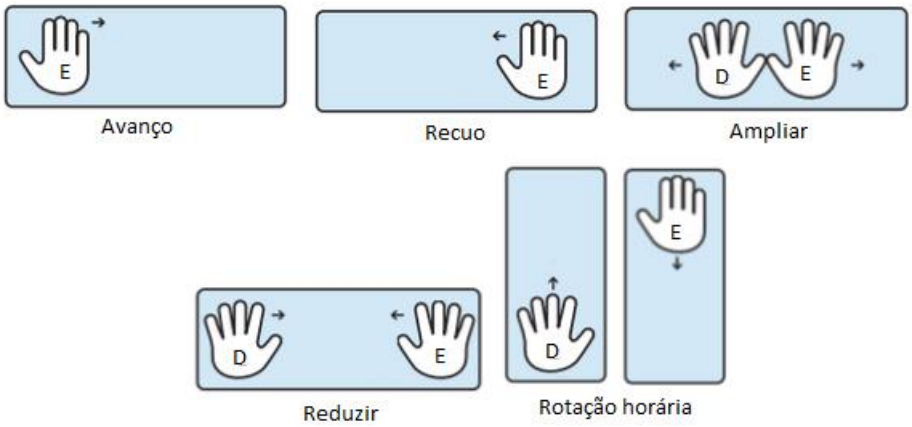


Figura 82. Alguns gestos em movimento.

Estes movimentos podem ser aplicados em diversas situações, como controlo de slides numa apresentação PowerPoint, no controlo do globo no programa Google Earth, controlar a visualização de um ficheiro 3D num programa de CAD, etc…

Para a classificação dos gestos da figura 82 apenas os valores resultantes das médias dos vetores quer verticais quer horizontais são contabilizados. Em que um movimento na horizontal será um vetor com um valor imaginário elevado e um valor real muito baixo e vice-versa para um movimento na vertical.

Para os gestos com as duas mãos basta ter em conta que enquanto uma mão terá um vetor com um valor positivo (horizontal ou vertical) a outra, se estiver a realizar o movimento no sentido contrário à mesma velocidade que a primeira mão, terá um valor negativo mas com módulo de velocidade igual.

Para o controlo do cursor do rato, os valores médios dos vetores serão sempre adicionados às últimas coordenadas conhecidas do cursor.

Como referido anteriormente, a utilização do *Optical Flow* deve-se muito ao nível de aplicação que desejamos e assim ter um sistema mais parecido com o funcionamento do rato ao invés do sistema de coordenadas do Tablet (coordenadas absolutas) tornando-se mais intuitivo e mais *user friendy*.

Contudo o sistema de coordenadas absolutas pode ser mais fiável em termos de exatidão do que a utilização do *Optical Flow*.

Para a utilização do *Optical Flow* o sistema desenvolvido obtém as coordenadas do cursor antes de fazer o processamento da imagem. Quando o utilizador fizer um movimento com a mão e esta estiver a fazer um gesto com a mão aberta e cinco dedos levantados, os vetores resultantes desse movimento vão adicionar às componentes das coordenadas respetivas da posição atual do cursor, ou seja, a componente horizontal do vetor de *Optical Flow* vai adicionar à coordenada x do cursor e a componente vertical do vetor adiciona à coordenada y do cursor. Isto vai causar uma alteração nas coordenadas (x,y) do cursor fazendo-o mover de acordo com o vetor resultante do movimento da mão. Por fim obtêm-se as coordenadas finais, que serão posteriormente adicionadas ao novo vetor de movimento.

A maior desvantagem do sistema de coordenadas incrementais usando os vetores obtidos pelo *Optical Flow* é o problema da abertura do diafragma, como já ilustrado. Num movimento da mão puramente horizontal vão aparecer vetores com componente vertical e como trabalhamos com as médias dos vetores resultantes quer horizontais quer verticais, acontece que num movimento horizontal teremos sempre uma componente vertical mesmo que seja muito pequena. Esse valor vertical será adicionado à coordenada y do cursor do rato fazendo-o andar na vertical, quando o que realmente se quer é um movimento apenas horizontal. Por este motivo o controlo do cursor do rato não será muito preciso, terá sempre flutuações, devido aos vetores residuais. Tal facto torna-se mais critico quando se pretende movimentar o cursor para botões mais pequenos, numa janela ou ambiente de trabalho.

Para minimizar os efeitos causados pelos vetores residuais, utilizou-se o algoritmo para o filtro de Kalman [34].

3.6 Aplicação do filtro de Kalman

O filtro de Kalman tem como principal aplicação a predição das coordenadas de objetos em movimento, mas também é usado para diminuir o ruido das coordenadas atuais de um objeto, muito aplicado em aplicações para radares [35]:

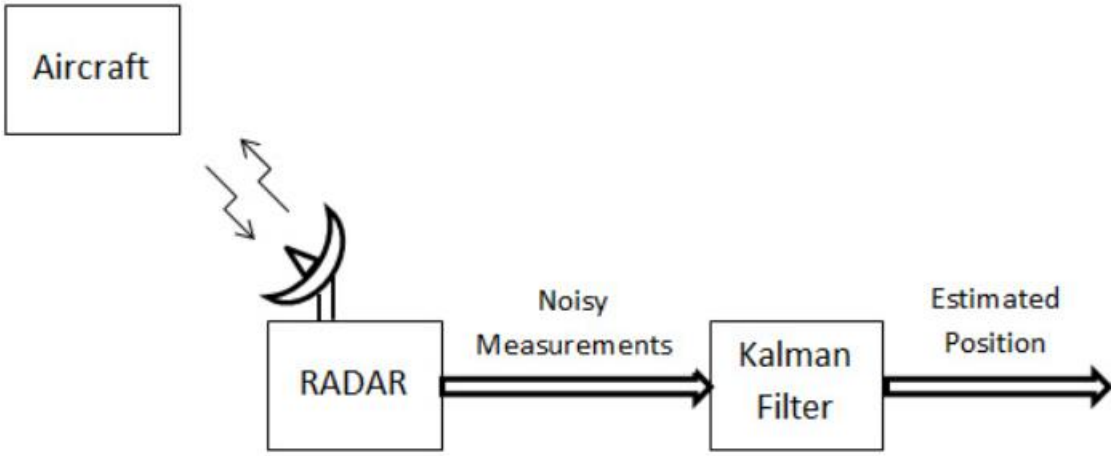


Figura 83. Exemplo de aplicação do filtro de Kalman para radares. Créditos MATLAB

Assim o filtro de Kalman reduz ou elimina os ruídos obtidos na determinação das coordenadas finais do cursor, assumindo que o objeto (neste caso a mão) se movimente com velocidade ou aceleração constante.

Usando o filtro de Kalman para eliminar o ruido quando a posição da mão está numa região quase estática obtém-se o seguinte exemplo:

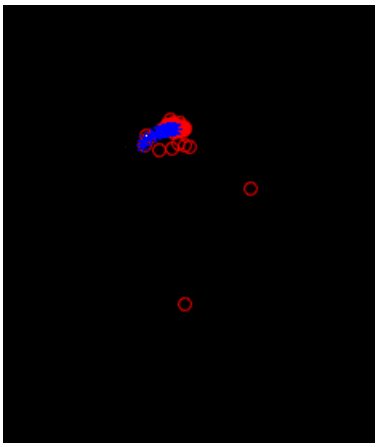


Figura 84. Eliminação do ruido usando o filtro de Kalman.

Devido ao problema da abertura do diafragma do *Optical Flow*, que resulta em vetores de ruido e à impossibilidade de um utilizador manter a mão numa posição completamente estática, surge a necessidade de filtrar os movimentos para uma maior exatidão nas posições do cursor. Na imagem da figura 84 a posição do centro da mão está representada pelos

círculos vermelhos. É possível verificar o ruido existente. (dois círculos até estão bastante distantes da maioria das outras posições).

Usando o filtro de Kalman é possível obter uma região mais bem definida com pouco ruido, como está representado pelos asteriscos azuis. Assim torna-se mais fácil obter uma localização mais precisa, sendo muito útil quando for necessário mover o cursor para botões muito pequenos.

3.7 Seguimento da trajetória

Como não queremos que todos os movimentos com a mão sejam interpretadas como gestos, nem que sejam interpretadas como movimentos para o cursor, é necessário definir umas formas de mão que sejam atuadores, por exemplo, para o movimento do cursor só é adicionado o vetor de movimento às coordenadas do cursor, sempre que a mão esteja aberta e com pelo menos 4 dedos levantados.

Para além dos gestos referidos acima, foram também criados gestos em movimento para fazer atuações ao programa, como por exemplo terminar a captura da imagem e fechar o programa de interação humano-computador.

Para a distinção dos gestos, o seu estudo foi feito a partir da trajetória que realizam. Se a mão numa determinada forma realizar um trajeto num limite de tempo, então o programa reconhece o trajeto e termina o ciclo de aquisição, desligando a conexão à Kinect.

O método criado para a distinção envolve a aquisição das coordenadas do centro da palma da mão e de uma divisão da imagem em regiões.

A imagem em profundidade da Kinect é representada num frame com dimensões de 640x480 pixéis. O frame é então dividido em 4 regiões na horizontal e 4 na vertical totalizando num total de 16 regiões. Na horizontal divide-se de 160 em 160 pixéis, e na vertical de 120 em 120 pixéis.

A numeração das regiões é importante para a determinação da trajetória, pois será a partir da numeração que a trajetória será definida.

A numeração das regiões é crescente da esquerda para a direita e de cima para baixo como representado na imagem da figura 85 [ver também figura 85].

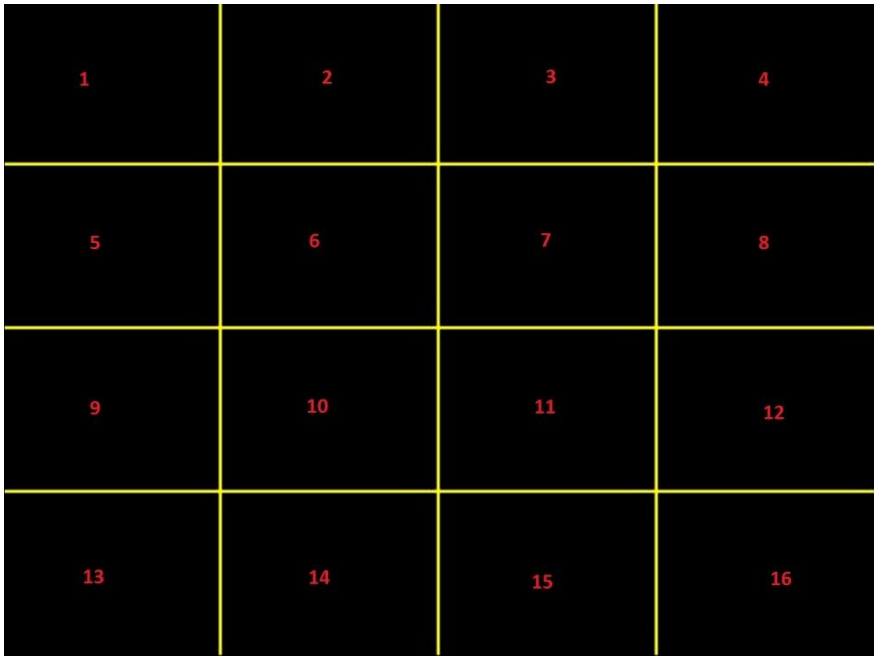


Figura 85. Representação das regiões e numeração respetiva.

Quando o utilizador estiver com a mão numa determinada forma, neste caso com 3 dedos levantados, as coordenadas atuais da palma da mão serão traduzidas na região respetiva, por exemplo:

Quando as coordenadas da mão sejam: X>320 & X<=480 & Y>120 & Y<=240

Então as coordenadas da mão serão guardada numa matriz como região 7.

A trajetória será traduzida pelas regiões em que o centro da palma da mão passa [ver figura 87].

Um exemplo de uma matriz resultante das coordenadas:

Traj=[7 7 7 11 11 10 10 10 10 10 10 9 9]:

Assim temos uma palma da mão que passou pelas regiões 7, 11, 10 e 9. O facto de termos mais dados na região 10 do que na região 9 não é importante, o que interessa são as regiões e a ordem pela qual a mão passa.

Se o utilizador desejar cancelar o seguimento (*tracking*) da trajetória depois de a ter iniciado, basta realizar outro gesto qualquer, como por exemplo abrir a mão levantando os 5 dedos.

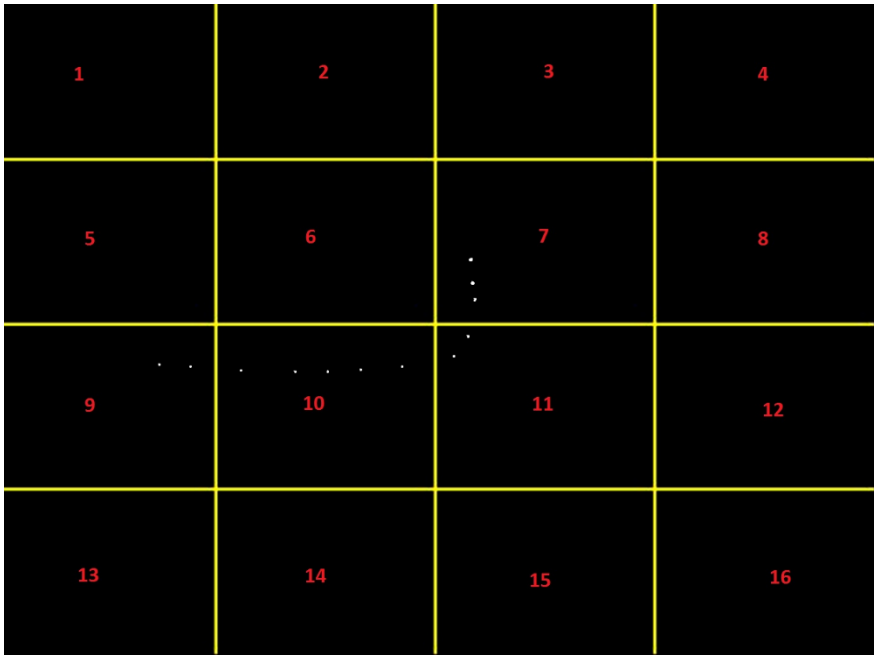


Figura 86. Exemplo de trajetória realizada.

Com o seguimento da posição efetuado, é necessário criar trajetórias para atuação. Uma das trajetórias mais intuitivas de realizar e de detetar é o sinal de "visto" ou V [ver figura 88].

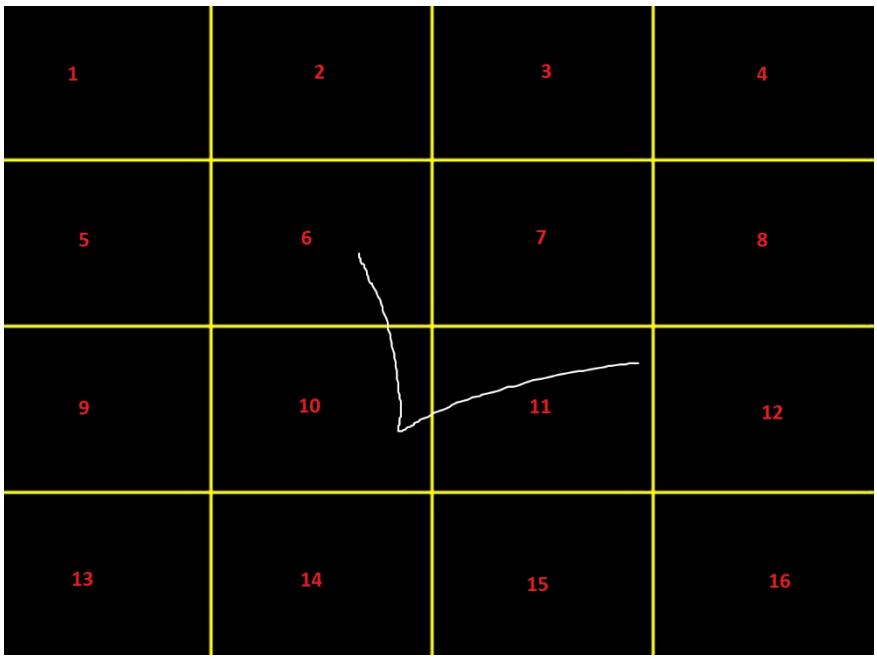


Figura 87. Realização de um visto para cancelar programa de aquisição de imagem.

Neste exemplo o "visto" que começa na região 6 passa por mais duas regiões: a 10 e 11. No entanto pode começar noutra região e passar por outras. É por isso que a ordem das regiões é importante. Como já referido o importante é saber em que região se começa a fazer o seguimento e por onde passa a mão, portanto podemos obter uma matriz que seja Traj=[6 6 6 6 10 10 10 10 10 10 11 11 11 11 11 11 11];

A matriz será transformada em TrajReduz=[6 10 11], com apenas a ordem das regiões por onde passa. Como a região inicial pode ser outra, desenvolveu-se a condição seguinte:

Se a segunda componente da matriz TrajReduz for igual ao valor da primeira componente + 4, e se a terceira componente for igual ao valor da primeira componente +5, então teremos um "visto" realizado. Assim o "visto" não necessita de começar na região 6 e passar pela 10 e 11, pode começar na 5 e passar por 9 e 10 ou começar na 9 e passar pela 13 e 14 etc…

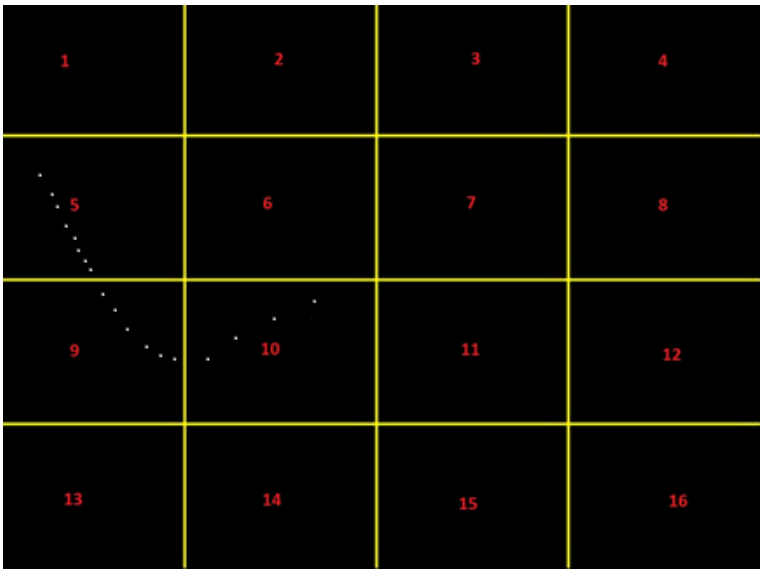


Figura 88. Exemplo de trajetória começando na região 5 e passando por 9 e 10.

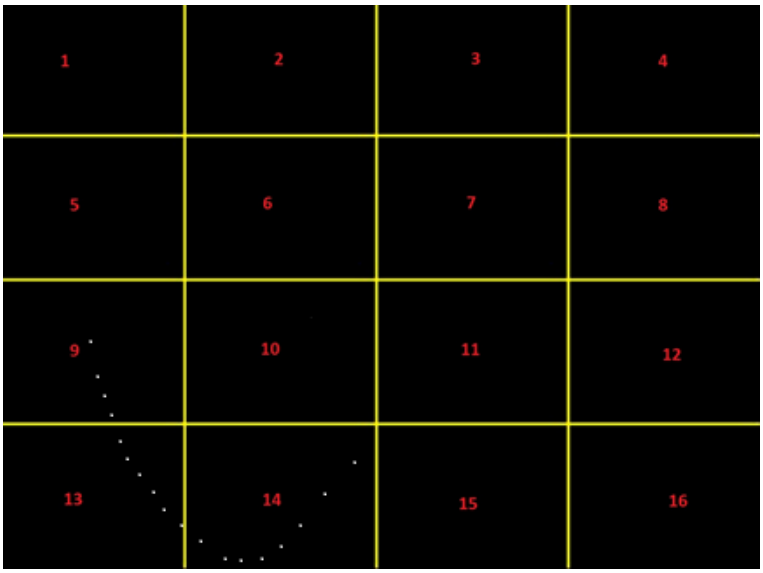


Figura 89. Exemplo de trajetória começando na região 9 e passando por 13 e 14.

3. 7 Atuação

Para a tradução dos gestos realizados em comandos interpretados pelo computador, o *Optical Flow* e a forma da mão incluindo o número de dedos levantados, foram os elementos estudados que permitirão fazer a distinção entre os gestos.

Para o controlo do movimento do cursor do rato, o princípio baseia-se na utilização do *Optical Flow* para a determinação dos vetores de movimento realizados pela mão.

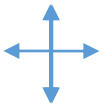
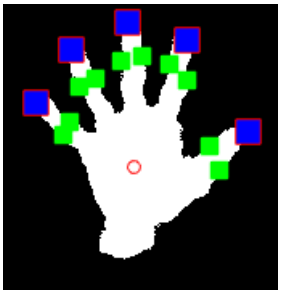
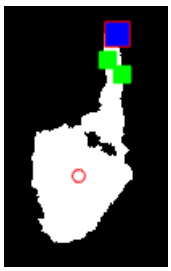
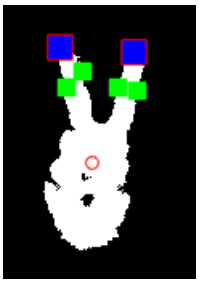
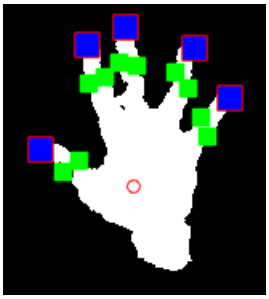
Para implementar as ações de *click* e duplo *click* foi necessário estudar a forma da mão associada ao número de dedos levantados.

Para a separação das diferentes formas de mão que se podem realizar utilizam-se três meios: Fator de forma, número de dedos detetados e por fim os momentos invariantes de imagem. Existem outos métodos, nomeadamente o *template matching* [36], que não foram estudados. No entanto estes podem oferecer mais robustez na distinção de gestos.

O método de *template matching* utiliza uma imagem de máscara/ *template* que será sobreposta em todos os pixéis de outra imagem com dimensões superiores e tentará encontrar o pixel na maior imagem onde a correlação seja máxima [37].

Os fatores de forma e os momentos invariantes de imagem são calculados para cada gesto *A priori* numa imagem binária com apenas o gesto pretendido. Os valores são guardados, e sempre que num fluxo de vídeo, um frame tiver presente uma imagem com um fator de forma e um momento invariante que pertença ao limite dos fatores e momentos previamente calculado, existe a possibilidade de estarmos perante um gesto conhecido sendo assim realizada a distinção de gestos estáticos.

Alguns gestos e seu propósito estão representados nas figuras seguintes.

Gesto Adquirido da câmara	Representação Binária	Função
  Mão direita		Controlo do Cursor
  Ambas as mãos		Movimento em vazio
 Ambas as mãos		*Click*
 Ambas as mãos		Duplo *Click*
 Mão esquerda		Controlo dos slides no PowerPoint: avançar ou retroceder slide.

Capítulo 4

Conclusões Discussão dos resultados

O programa e os algoritmos desenvolvidos durante a realização da dissertação cumpriram o objetivo inicial proposto. O controlo de apresentações em PowerPoint com recurso a gestos adquiridos por uma câmara 3D foi um sucesso. Contudo, existe muita margem para melhoramentos e até mesmo a extrapolação do programa desenvolvido para diversas aplicações de interação humano-computador, como por exemplo videojogos.

Como foi mencionado anteriormente, o código foi desenvolvido em MATLAB, apesar do *software* ser bastante bom para a análise dos algoritmos, em termos de tempo de processamento é significativamente mais lento que outras linguagens de programação.

O tempo de processamento de cada frame foi desde o início uma grande preocupação. Tentar fazer com que o processamento fosse em tempo real foi um dos maiores desafios. Optou-se então por algoritmos que exigiam menores recursos e cálculos mais simples, o que por vezes pode aumentar o erro em relação a algoritmos mais desenvolvidos, como é o caso do algoritmo desenvolvido para determinar o centro da palma da mão.

Neste capítulo explica-se como é possível controlar, com uma certa limitação, uma apresentação PowerPoint e o cursor do rato com recurso a gestos, mais o gesto de término de aquisição de imagem e processamento dos gestos.

4.1 Segmentação

A segmentação das mãos foi o primeiro passo a ser desenvolvido. A capacidade de obtenção de uma mão bem definida é de extrema importância para o processamento da forma da mão de maneira correta.

A utilização combinada entre a imagem a cores e o algoritmo de reconhecimento de pele e a imagem em profundidade da Kinect resultam numa imagem binária onde é possível estudar a forma da mão bem como os movimentos efetuados.

Contudo a segmentação nem sempre é perfeita. Quando ocorre uma falha durante a aquisição de imagem, quer pela imagem a cores quer pela imagem em profundidade, vão ser criadas falhas na imagem binária de estudo, impossibilitando assim um estudo correto. Ocasionalmente existem certas falhas como regiões no centro da mão que apresentam buracos [ver figura 32], ou dedos que parecem unidos.

Algumas das causas para as falhas de segmentação podem surgir de erros na determinação dos padrões de infravermelhos criados pela câmara 3D, como por exemplo a interferência causada por outras fontes de radiação infravermelha como a luz solar. Outras falhas podem partir de erros na determinação de pixéis da cor da pele, e ainda o posicionamento errado perante a câmara [ver figura 30 e 31].

4.2 Determinação do centro da palma da mão.

A determinação do centro da palma da mão a partir do cálculo das distâncias dos pontos da mão à sua fronteira é um método fiável, mas que pode ser muito pesado em termos computacionais [ver tabela 1]. Existem outros métodos para a determinação do centro da palma da mão que não seja pela determinação do centroide do objeto. A criação de uma *bounding box* que envolvesse a mão e a determinação do centro dessa mesma caixa foi um método considerado mas sem os resultados pretendidos.

Portanto, foi necessário reduzir os números de distâncias a ser calculadas para tornar o processamento mais rápido, sem comprometer o erro na aquisição do ponto central da palma da mão. Como a forma da mão está em constante mudança, encontrar um número correto de pontos a serem analisados quer para a fronteira quer para o interior da mão não é fácil. No entanto para o programa feito os pontos interiores são calculados de 75 em 75 e os pontos fronteiros de 10 em 10. Em qualquer instante a posição do centro da palma da mão é calculada independentemente da forma que a mão assuma.

4.3 Contagem dos dedos levantados.

A determinação da localização das pontas dos dedos e consequente contagem dos que estão levantados foi combinada com os pontos obtidos pelo algoritmo Convex Hull associados à K-Curtvature que, apesar de ser uma combinação muito eficaz, não está imune a erros. Esses erros derivam da obtenção do polígono convexo que engloba a mão como é o caso verificado na figura 54. A falha é mais evidenciada caso o utilizador esteja com os cinco dedos levantados. Com apenas dois ou três dedos levantados a probabilidade falhar a determinação dos dedos é menor.

Realizando alguns testes, contando o número de frames em que o número de dedos e a sua posição são detetadas corretamente obteve-se a seguinte tabela de eficácia:

Tabela 4. Tabela de eficácia para determinação de dedos levantados.

Gesto realizado	Percentagem de frames. Identificação e deteção correta do número de dedos
	96%
	65%
	94%
	82%
	96%
	85%

A aplicação da alternativa desenvolvida [ver figura 55], ou seja, com a determinação do ângulo de curvatura a todos os pontos fronteiros da mão, torna-se muito trabalhosa em termos de número de cálculos a ser realizados, aumentando assim o tempo de processamento do frame.

4.4 Distinção entre mão esquerda e a direita.

Para a distinção entre a mão esquerda e mão direita, a metodologia escolhida está limitada pela capacidade de encontrar os dedos polegares e indicadores de cada mão. Portanto, a distinção entre as mãos não será feita com qualquer forma, como demonstrado nas figuras 74 e 75.

Determinados os três pontos necessários, o ponto correspondente ao dedo indicador, o ponto do dedo polegar e o ponto central da palma da mão, retiram-se os dois vetores necessários, cada um entre um dedo e o centro da palma, para o cálculo do vetor externo.

Com o algoritmo desenvolvido é possível distinguir as mãos, mesmo que os braços estejam cruzados ou que apareça apenas uma mão na imagem.

Todavia, o programa não é infalível, já que existe a possibilidade na determinação do vetor polegar estar representada no dedo indicador e o vetor indicador estar representado para o dedo polegar Nesses casos, com os dedos identificados incorretamente, o resultado será dado de forma contrária à mão. Ou seja, a identificação dos dedos trocada, troca a identificação da mão. Estes erros podem ocorrer por causa da região entre os dedos determinada ser errada, ou mais frequentemente, a distância do centro da palma da mão ao dedo indicador ser menor do que da distância ao dedo polegar.

Com a mão aberta e os 5 dedos levantados, e a mão em constante movimento a eficácia do algoritmo desenvolvido foi de 85%.

4.5 Optical Flow e gestos em movimento

A utilização do Optical Flow foi dos desafios mais interessantes no desenvolvimento da dissertação e do programa.

A aplicação no programa foi bem-sucedida e permitiu com bastante facilidade a interação utilizador-computador. A implementação do filtro de Kalman no Optical Flow resultou em movimentos mais precisos.

Certos gestos como a rotação, avanço/recuo e Zoom, são feitos com recurso à combinação entre a distinção de mãos e o Optical Flow, visto que é possível distinguir o movimento para cada mão separadamente.

Ao contrário do que se esperava, o cálculo do Optical Flow de um movimento não exige uma grande capacidade computacional, verificando-se tempos de processamento muito rápidos.

O Optical Flow ainda apresenta o problema da abertura do diafragma que diminui a precisão dos movimentos. Apesar do filtro de Kalman conseguir minimizar esse efeito, ele ainda está presente em todos os movimentos.

Para gestos mais complexos, com instruções mais específicas, o sistema de divisão da imagem em regiões e a análise da trajetória realizada permite o estudo de gestos em movimento mais complexos, que seriam mais dificilmente analisados caso fosse utilizado o Optical Flow. Apesar de ter sido apenas criado um gesto (o sinal de "visto" para terminar a aquisição de imagem), o sistema pode ser utilizado para gestos com trajetórias mais complexas.

Observações finais

Os testes realizados permitiram perceber como a utilização dos algoritmos se comportam em diferentes condições, como por exemplo para a segmentação, sendo importante seguir as condições referidas. A obrigatoriedade das palmas da mão estarem apontadas para a câmara limita o trajeto que a mão pode fazer. Outras limitações envolvem a capacidade do programa em detetar todos os dedos levantados, quer devido à segmentação, quer à aplicação do algoritmo de Convex Hull. Apesar de conterem algumas falhas, quer o sistema de determinação de dedos levantados quer a distinção entre mão esquerda e direita, conseguem obter resultados aceitáveis.

A utilização do Optical Flow e do filtro de Kalman funcionou de forma bastante fluida e intuitiva, permitindo a criação de comandos fáceis de usar.

É de salientar que o programa desenvolvido cumpre os objetivos inicialmente propostos, sendo possível controlar programas como o PowerPoint usando um sistema de visão artificial. O tempo de processamento de cada frame, apesar de ser mais demorado do que se pretendia (cerca de 0.6-0.7 segundos por frame), permite o controlo do computador em tempo real.

Os algoritmos desenvolvidos podem ser adaptados com facilidade a outras plataformas de interação Humano-Computador mais adequadas como a NUI (natural user interface). Contudo, alguns algoritmos necessitam de ser trabalhados, e será necessário ainda estudar outras metodologias, como a já mencionada *Template Matching*, e a criação de gestos com trajetórias mais complexas.

Trabalho futuro

O programa desenvolvido estuda as mãos de forma sequencial, ou seja, só processa a segunda mão depois da primeira estar terminada. Torna-se assim necessário criar um

programa que processe as duas mãos em simultâneo, diminuindo o tempo total de processamento.

Para trabalho futuro, considera-se necessário implementar os algoritmos desenvolvidos em linguagem C++ ou C# com recurso a bibliotecas como OpenCV e OpenNI.

Uma melhoria do sistema de deteção dos dedos será também de extrema importância. Uma alternativa ao Convex Hull poderia ser bastante vantajoso, apesar do sistema K-Curvature ser bastante fiável.

A distinção entre a mão esquerda e a mão direita pode ser melhorada. A determinação dos vetores polegar e vetores indicador podem ser feitos com recurso a outros pontos, o que pode diminuir o erro.

O reconhecimento das formas da mão foi realizado utilizando descritores, como o fator de forma e os momentos invariantes. No futuro seria interessante fazer o reconhecimento através de *features extraction* ou *template matching* de forma a estudar as vantagens.

Bibliografia

[1] "Human Computer Interaction - brief intro." [Online]. Available: https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed/human-computer-interaction-brief-intro. [Accessed: 07-Oct-2015].

[2] "Galeria | games500.hol.es." [Online]. Available: http://games500.hol.es/?page_id=69. [Accessed: 07-Oct-2015].

[3] "Google juices Chrome OS with fondleslab smarts • The Register." [Online]. Available: http://www.theregister.co.uk/2011/04/07/chrome_os_for_tablets/. [Accessed: 07-Oct-2015].

[4] "Speech Recognition Through the Decades: How We Ended Up With Siri | TechHive." [Online]. Available: http://www.techhive.com/article/243060/speech_recognition_through_the_decades_how_we_ended_up_with_siri.html. [Accessed: 07-Oct-2015].

[5] "Microsoft HoloLens | Official Site." [Online]. Available: https://www.microsoft.com/microsoft-hololens/en-us. [Accessed: 07-Oct-2015].

[6] "Samsung Gear." [Online]. Available: http://www.samsung.com/global/microsite/gearvr/gearvr_features.html. [Accessed: 07-Oct-2015].

[7] "Gamasutra - In-Depth: Eye To Eye - The History Of EyeToy." [Online]. Available: http://www.gamasutra.com/php-bin/news_index.php?story=20975. [Accessed: 07-Oct-2015].

[8] Microsoft Corporation, "Kinect for Windows | Human Interface Guidelines v1.8," pp. 1–142, 2013.

[9] "Motion Capture Software and Mocap Tracking Info." [Online]. Available: http://www.organicmotion.com/motion-capture/. [Accessed: 07-Oct-2015].

[10] "How Microsoft Kinect changed technology - CNN.com." [Online]. Available: http://edition.cnn.com/2011/11/11/tech/how-microsoft-kinect-changed-technology/index.html. [Accessed: 07-Oct-2015].

[11] "Kinect hardware." [Online]. Available: https://dev.windows.com/en-us/kinect/hardware. [Accessed: 07-Oct-2015].

[12] *Intelligent Autonomous Systems 12: Volume 1: Proceedings of the 12th International Conference IAS-12, Held June 26-29, 2012, Jeju Island, Korea*, vol. 2. Springer Science & Business Media, 2012.

[13] "PHOTONIC FRONTIERS: GESTURE RECOGNITION: Lasers bring gesture recognition to the home - Laser Focus World." [Online]. Available: http://www.laserfocusworld.com/articles/2011/01/lasers-bring-gesture-recognition-to-the-home.html. [Accessed: 07-Oct-2015].

[14] "Kinect Pattern Uncovered | azt.tm's Blog." [Online]. Available: https://azttm.wordpress.com/2011/04/03/kinect-pattern-uncovered/. [Accessed: 07-Oct-2015].

[15] K. Khoshelham, "Accuracy Analysis of Kinect Depth Data," *ISPRS - Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.*, vol. XXXVIII–5/, no. August, pp. 133–138, 2012.

[16] "Skeletal Tracking." [Online]. Available: https://msdn.microsoft.com/en-us/library/hh973074.aspx. [Accessed: 21-Oct-2015].

[17] "Software fisioterapia Kinect - Jintronix." [Online]. Available: http://www.jintronix.com/pt/. [Accessed: 21-Oct-2015].

[18] "mit-ros-pkg - ROS ." [Online]. Available: http://wiki.ros.org/mit-ros-pkg. [Accessed: 21-Oct-2015].

[19] "kinect - ROS ." [Online]. Available: http://wiki.ros.org/kinect. [Accessed: 21-Oct-2015].

[20] "Project Soli." [Online]. Available: https://www.google.com/atap/project-soli/. [Accessed: 21-Oct-2015].

[21] J. V. e C. J. D. L. Basmajian, *Muscles alive: their functions revealed by electromyography*. Williams & Wilkins, 1985.

[22] Barbosa, Rui Pedro Anastácio - "Reconhecimento de gestos para apresentações informáticas interativas," 2014. Tese de Mestrado.

[23] R. Lockton, "Hand Gesture Recognition Using Computer Vision," *4th Year Proj. Report-Oxford Univ.*, pp. 1–69, 2002.

[24] H. S. Yeo, B. G. Lee, and H. Lim, "Hand tracking and gesture recognition system for human-computer interaction using low-cost hardware," *Multimed. Tools Appl.*, pp. 1–29, 2013.

[25] Z. Ren, J. Yuan, and Z. Zhang, "Robust hand gesture recognition based on finger-earth mover's distance with a commodity depth camera," *Proc. 19th ACM Int. Conf. Multimed.*, pp. 1093–1096, 2011.

[26] T. Liu, A. W. Moore, A. Gray, and K. Yang, "An investigation of practical approximate nearest neighbor algorithms," *Adv. Neural Inf. Process. Syst.*, p. 8, 2004.

[27] "CGAL 4.7 - 2D Convex Hulls and Extreme Points: User Manual." [Online]. Available: http://doc.cgal.org/latest/Convex_hull_2/. [Accessed: 25-Oct-2015].

[28] D. Lee, "Vision-Based Finger Action Recognition by Angle Detection and Contour Analysis," *ETRI J.*, vol. 33, no. 3, pp. 415–422, 2011.

[29] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," *Int. J. Comput. Vis.*, vol. 92, no. 1, pp. 1–31, 2011.

[30] "Estimate object velocities - MATLAB." [Online]. Available: http://www.mathworks.com/help/vision/ref/vision.opticalflow-class.html. [Accessed: 27-Oct-2015].

[31] “Optical Flow Constraint Equation.” [Online]. Available: https://www.cs.cf.ac.uk/Dave/Vision_lecture/node47.html. [Accessed: 27-Oct-2015].

[32] B. K. P. Horn and B. G. Schunck, “Determining optical flow,” *Artif. Intell.*, vol. 17, no. 1–3, pp. 185–203, 1981.

[33] N. Recipes, “Motion Estimation Why estimate motion ? Optical flow Problem definition : optical flow Optical flow constraints (grayscale images) Optical flow equation.”

[34] R. E. Kalman and Others, “A new approach to linear filtering and prediction problems,” *J. basic Eng.*, vol. 82, no. 1, pp. 35–45, 1960.

[35] F. Svanstrom, “Kalman Filtering,” Linnaeus University.

[36] R. Brunelli, “Template matching techniques in computer vision: theory and practice,” *Theory Pract.*, p. 338, 2009.

[37] “Template Matching — OpenCV 2.4.12.0 documentation.” [Online]. Available: http://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/template_matching/template_matching.html. [Accessed: 04-Nov-2015].