Reconstrução de Superfícies usando um Scanner 3D de Luz Estruturada

Nuno Teófilo Pereira Araújo Fernandes

Dissertação submetida ao Instituto Superior de Engenharia do Porto para a obtenção do grau de Mestre em Engenharia de Computação e Instrumentação Médica

Orientador:
Doutor Carlos Vinhais
Departamento de Física do ISEP

Porto, 11 de Julho de 2012

"Para ser grande, sê inteiro: nada
Teu exagera ou exclui.
Sê todo em cada coisa. Põe quanto és
No mínimo que fazes.
Assim em cada lago a lua toda
Brilha, porque alta vive."

Ricardo Reis, in "Odes"
Heterónimo de Fernando Pessoa

Agradecimentos

Gostaria de apresentar os meus sinceros agradecimentos ao meu orientador, Prof. Doutor Carlos Vinhais, pela ajuda, motivação e paciência com que me brindou ao longo do desenvolvimento deste trabalho.

À Patrícia Pereira apresento o meu profundo agradecimento, porque ao longo do desenvolvimento deste trabalho a sua presença e a sua paciência foram constantes, apoiando-me e suportando-me incondicionalmente. Sem o seu apoio, não teria conseguido.

Ao Ricardo Pinho, o meu obrigado por todo o apoio, e pelo desejo sincero de que este trabalho se concretizasse.

À minha família, eu apresento a minha gratidão pela enorme força e pelo suporte que me tem dado desde que me lembro.

A todos os demais colegas e amigos que me apoiaram e encorajaram nos momentos de desmotivação, que me deram ideias e partilharam conhecimento comigo, o meu muito obrigado.

Resumo

Ao longo dos últimos anos, os *scanners* 3D têm tido uma utilização crescente nas mais variadas áreas. Desde a Medicina à Arqueologia, passando pelos vários tipos de indústria, é possível identificar aplicações destes sistemas. Essa crescente utilização deve-se, entre vários factores, ao aumento dos recursos computacionais, à simplicidade e à diversidade das técnicas existentes, e às vantagens dos *scanners* 3D comparativamente com outros sistemas. Estas vantagens são evidentes em áreas como a Medicina Forense, onde a fotografia, tradicionalmente utilizada para documentar objectos e provas, reduz a informação adquirida a duas dimensões.

Apesar das vantagens associadas aos *scanners* 3D, um factor negativo é o preço elevado. No âmbito deste trabalho pretendeu-se desenvolver um *scanner* 3D de luz estruturada económico e eficaz, e um conjunto de algoritmos para o controlo do *scanner*, para a reconstrução de superfícies de estruturas analisadas, e para a validação dos resultados obtidos. O *scanner* 3D implementado é constituído por uma câmara e por um projector de vídeo *"off-the-shelf"*, e por uma plataforma rotativa desenvolvida neste trabalho. A função da plataforma rotativa consiste em automatizar o *scanner* de modo a diminuir a interacção dos utilizadores. Os algoritmos foram desenvolvidos recorrendo a pacotes de *software open-source* e a ferramentas gratuitas.

O *scanner* 3D foi utilizado para adquirir informação 3D de um crânio, e o algoritmo para reconstrução de superfícies permitiu obter superfícies virtuais do crânio. Através do algoritmo de validação, as superfícies obtidas foram comparadas com uma superfície do mesmo crânio, obtida por tomografia computorizada (TC). O algoritmo de validação forneceu um mapa de distâncias entre regiões correspondentes nas duas superfícies, que permitiu quantificar a qualidade das superfícies obtidas.

Com base no trabalho desenvolvido e nos resultados obtidos, é possível afirmar que foi criada uma base funcional para o varrimento de superfícies 3D de estruturas, apta para desenvolvimento futuro, mostrando que é possível obter alternativas aos métodos comerciais usando poucos recursos financeiros.

Abstract

Over the past few years, 3D scanners have experienced an increasing use in various areas. From Medicine to Archeology, passing through different types of industry, it is possible to identify applications of these systems. This increasing use is due, among many other factors, to the evolution of computing resources, the simplicity and diversity of techniques, and the advantages of 3D scanners when compared to other systems. These advantages are clear in areas as Forensic Medicine, where 2D photography, commonly used in object and evidence documentation, reduces the acquired information to two dimensions.

Despite the advantages of 3D scanners, their high price is considered a drawback. In this work one intended to develop an economic and effective 3D structured light scanner, as well as a set of algorithms to control the scanner, to reconstruct surfaces from analised structures, and to validate the obtained results. The 3D scanner is formed by a camera and a video projector, both "off-the-shelf"components, and a turnable platform. The platform's main function is to automate the scanner in order to decrease user interaction. The algorithms were developed using open-source software packages and free development tools.

The 3D scanner was then used to analise a skull, and the reconstruction algorithm allowed a virtual surface of the skull to be reconstructed. Through the validation algorithm, the obtained surfaces were compared with a reference surface of the same skull, obtained using a Computerized Tomography scanner. In the end, the validation algorithm provided a distance map between corresponding regions on both surfaces, which allowed to quantify the virtue of the reconstructed surfaces.

Considering the presented work and the obtained results, one can afirm that a functional basis to acquire 3D information and to reconstruct 3D surfaces was succefully created. All of the presented work allows further development and shows that it is possible to obtain alternatives to commercial devices, using little financial resources.

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Enquadramento

Ao longo dos últimos anos tem-se verificado um aumento na diversidade de técnicas para a aquisição de informação tridimensional de estruturas e para a reconstrução das respectivas superfícies 3D. Este facto deve-se, em grande parte, aos avanços nos recursos computacionais e informáticos, e ao elevado número de aplicações em diversas áreas como a Medicina, a Arqueologia, a Indústria Cinematográfica, entre outras [1]. Uma área de grande importância para a sociedade, na qual se tem verificado uma utilização crescente deste tipo de tecnologia, é a área da Medicina Forense [2]. A substituição da fotografia tradicional por modelos 3D tem permitido documentar com mais detalhe estruturas anatómicas, traumas em vítimas, objectos e pistas. Com o auxílio de aplicações computacionais e partindo destes modelos, é possível, por exemplo, verificar a correspondência entre a forma de um objecto e o trauma numa vítima, permitindo concluir se foi ou não o objecto que causou o trauma [3]. Com modelos 3D e sistemas de prototipagem, também é possível originar modelos físicos com as dimensões do modelo original. Desta forma, a estrutura do modelo original é preservada, caso sejam realizados testes que comprometam as suas propriedades físicas [4]. Geralmente, para se adquirir informação 3D de estruturas anatómicas, recorre-se à Tomografia Computorizada (TC) [2]. Este método permite adquirir com elevado detalhe e precisão a forma das estruturas em estudo. No entanto, além do *scanner* de TC ser um equipamento caro, emite radiação ionizante. Uma alternativa para adquirir informação 3D de estruturas, quando só se pretende informação da superfície externa, consiste na utilização de *scanners* 3D não baseados na emissão de radiação ionizante [5]. *Scanners* 3D por contacto e

scanners 3D ópticos reflexivos são boas alternativas à TC, constituindo meios mais económicos e menos perigosos para a aquisição de informação 3D. A partir da informação obtida por estes sistemas, é possível proceder à reconstrução de superfícies 3D das estruturas analisadas.

1.2 Objectivos e Motivação

A aquisição de informação 3D e a reconstrução de superfícies constituem ainda hoje um desafio, existindo vários factores a considerar no desenvolvimento de métodos para esse fim. Esses factores compreendem o custo financeiro, o tipo de estrutura a analisar, a informação que se vai obter (só a estrutura ou também a textura, por exemplo), e a qualidade pretendida do modelo final. Existem no mercado vários sistemas para a aquisição de informação 3D e para a reconstrução de superfícies. No entanto, estes sistemas geralmente são bastante caros e, dependendo da utilização pretendida, muitos possuem um conjunto de características desnecessárias, que aumentam o preço final. Considerando estes aspectos, pretendeu-se desenvolver um sistema económico e eficaz, que pudesse ser utilizado na área da Medicina Forense para a análise e documentação de estruturas anatómicas. Uma vez que na área da Medicina Forense é frequentemente necessário analisar estruturas sensíveis, optar pelo desenvolvimento de um sistema baseado em contacto não seria a melhor opção. Não só porque o estabelecimento de contacto nem sempre é possível, mas também porque pode comprometer a integridade física das estruturas em análise. Por outro lado, optar por um sistema óptico traria algumas vantagens, já que estes sistemas são mais rápidos e, como não requerem contacto com a estrutura a analisar, podem ser utilizados com estruturas sensíveis. Um *scanner* 3D baseado na projecção de luz estruturada constituiria uma boa solução, uma vez que é um sistema eficaz, simples, e bem documentado. As vantagens desde tipo de *scanner* 3D, bem como as suas principais características serão aprofundadas ao longo deste documento. Assim, no âmbito deste trabalho foi desenvolvido um *scanner* 3D activo baseado na projecção de luz estruturada. Foi também desenvolvido um conjunto de algoritmos para a aquisição de informação 3D, para a reconstrução das respectivas superfícies 3D, e para a validação dos resultados obtidos. Este último passo - a validação dos resultados obtidos -, foi fundamental e conseguiu-se através da comparação das superfícies obtidas com uma superfície de referência obtida por TC. O *scanner* 3D é constituído por um projector de vídeo, por uma câmara, e também por uma plataforma rotativa para diminuir a interação dos utilizadores. O *scanner* 3D é composto por dispositi-

vos comerciais económicos e elementos desenvolvidos no âmbito deste trabalho, e o código foi implementado utilizando meios gratuítos e *open-source*.

1.3 Contribuições

O *scanner* 3D e os algoritmos desenvolvidos neste trabalho constituem um método simples e económico para reconstruir uma superfície 3D de um objecto em estudo. O contributo deste trabalho é multidisciplinar:

- O *scanner* 3D implementado é constituído por um projector, por uma câmara, e por uma plataforma rotativa totalmente desenvolvida neste trabalho. Graças à plataforma rotativa, o *scanner* 3D requer pouca interacção dos utilizadores, analisando a superfície de um objecto de estudo em 360º de forma automática;

- Os algoritmos desenvolvidos permitem a calibração do *scanner* 3D, a aquisição de pontos tirando partido da plataforma rotativa, e a criação das respectivas superfícies 3D;

- O algoritmo desenvolvido para a validação das superfícies obtidas, permite avaliar a qualidade de superfícies 3D reconstruídas, comparando-as com uma superfície originada por TC. Este é um passo fundamental para se verificar a qualidade do modelo final obtido.

Através do trabalho desenvolvido, obteve-se assim uma base funcional para a construção de superfícies 3D de objectos, apta para desenvolvimento futuro, mostrando que é possível obter alternativas aos métodos comerciais usando menos recursos financeiros.

1.4 Organização da Tese

Este documento encontra-se organizado em capítulos que descrevem as diferentes fases do trabalho realizado:

- **Capítulo 2 - *Scanners 3D***
  São abordados os princípios fundamentais dos diferentes tipos de *scanners* 3D, focando os respectivos métodos de funcionamento, as suas vantagens e desvantagens. É também feita uma referência a aplicações dos *scanners* 3D nas diversas áreas da indústria e a alguns sistemas disponíveis no mercado.

- **Capítulo 3 - *Triangulação Óptica***
  Neste capítulo é feita uma breve revisão do princípio da triangulação, que permite obter a profundidade de pontos no espaço 3D. É também apresentado o modelo *pinhole*, utilizado para representar geometricamente uma câmara e um projector.

- **Capítulo 4 - *Implementação de um Scanner 3D***
  Descrevem-se as várias etapas de desenvolvimento do *scanner* 3D, as suas componentes de *hardware* e de *software*, sendo também apresentada a montagem experimental.

- **Capítulo 5 - *Aquisição de Nuvens de Pontos***
  É abordada a calibração do *scanner* 3D e descrito o procedimento utilizado para a aquisição de uma ou de várias nuvens de pontos. São apresentados e discutidos os resultados obtidos, sendo apresentados algoritmos de pré-processamento das nuvens de pontos obtidas.

- **Capítulo 6 - *Reconstrução de Superfícies***
  Neste capítulo são descritos os métodos adoptados para reconstruir superfícies a partir das nuvens de pontos obtidas pelo *scanner* 3D, e para a validação das superfícies geradas.

- **Capítulo 7 - *Conclusão***
  Apresentam-se conclusões finais sobre o trabalho desenvolvido e sobre os resultados obtidos. São discutidos aspectos relevantes do trabalho e são feitas sugestões de trabalho futuro.

Capítulo 2

Scanners 3D

2.1 Princípios dos *Scanners* 3D

Um *scanner* 3D é um dispositivo que analisa, de forma automática ou semi automática, um objecto do mundo real para obter informação sobre a sua forma tridimensional. À saída, fornece nuvens de pontos geométricos da superfície do objecto analisado [6]. A partir dessas nuvens de pontos, e através de processamento por *software*, é possível reconstruir um modelo virtual a três dimensões da superfície do objecto analisado. Alguns *scanners* 3D podem ainda recolher informações sobre a cor e textura do objecto. Os *scanners* 3D são utilizados em diversas áreas como Arte e Arqueologia, indústria do entretenimento, Metrologia, navegação autónoma, indústria moderna e Medicina. Ao longo dos últimos anos tem-se verificado um avanço significativo na tecnologia 3D, tanto a nível de *software* como de *hardware*, o que tem permitido analisar com elevada precisão a forma e as características da superfície de objectos [7]. Os *scanners* 3D podem ser divididos em dois grandes grupos: *scanners* de contacto e *scanners* sem contacto (do inglês, *range finders*) [8]. Tendo em conta o foco deste trabalho, será dado destaque aos *scanners* 3D sem contacto. A figura 2.1 apresenta uma taxonomia dos vários sistemas para aquisição de informação tridimensional da superfície de objectos.

2.1.1 Métodos por Contacto

Os *scanners* de contacto utilizam uma sonda analógica de diâmetro conhecido, que se move sobre a superfície do objecto em análise, e adquire pontos dessa superfície através do contacto. Um exemplo deste tipo de sistemas é o pantógrafo de Soren-

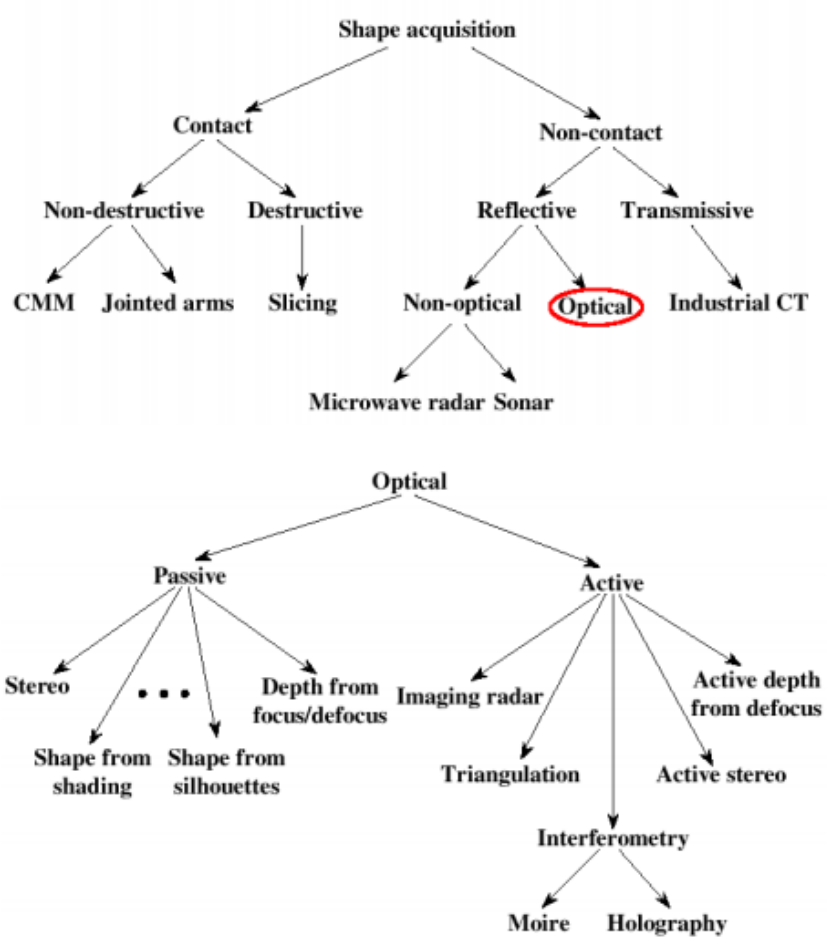
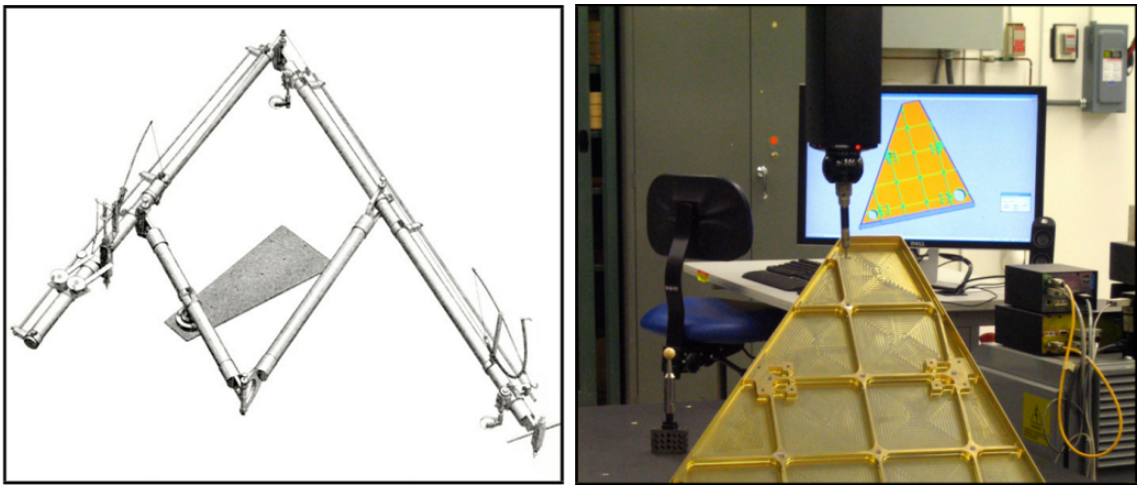


Fig. 2.1: Taxonomia dos vários sistemas para aquisição da forma 3D de objectos [9].

son patenteado em 1987 e o CMM (sistema de medição de coordenadas moderno), visíveis na figura 2.2 [10]. Estes sistemas têm algumas vantagens associadas, nomeadamente:

- Dispensam qualquer tipo de tratamento sobre a superfície do objecto para evitar reflexões;
- A densidade de dados não é fixa, mas depende da complexidade da superfície;
- Têm uma elevada precisão [11].

No entanto, o facto destes *scanners* necessitarem de contacto com o objecto torna-os inviáveis para certas aplicações, já que o contacto pode comprometer a integridade do material em análise [3]. Um outro aspecto a salientar é a lentidão dos *scanners* de contacto comparativamente com os *scanners* sem contacto [12].



(a) (b)

Fig. 2.2: (a) Pantógrafo de Sorenson. (b) CMM moderno. Adaptado de [10].

2.1.2 Métodos sem Contacto

Os *scanners* 3D sem contacto, tal como o nome indica, não estabelecem qualquer contacto com o objecto. Estes sistemas subdividem-se em transmissivos e reflexivos (ver figura 2.1). Como sistema transmissivo tem-se o equipamento de Tomografia Computorizada (TC), que se baseia na emissão e na detecção de radiação ionizante (raios-X), para a construção de imagens (aprofundado na secção 2.4). Dentro dos sistemas reflexivos, existem os sistemas não-ópticos e ópticos. Como exemplo de sistemas não-ópticos tem-se o sonar e o radar (*RAdio Detecting And Ranging*) de microondas. Estes sistemas determinam a distância a um objecto, emitindo um impulso de som ou de energia microondas, e medindo o tempo que este impulso demora a voltar do objecto. Por sua vez, os sistemas ópticos estão divididos em activos e passivos, englobando vários métodos para adquirir a forma tridimensional de objectos [9]. Na figura 2.3 é possível observar um conjunto de *scanners* 3D ópticos, bem como um equipamento de TC. Considerando o objectivo deste trabalho, será feita uma abordagem mais detalhada aos sistemas ópticos em geral, e aos sistemas baseados em triangulação.

Scanners ópticos

Os *scanners* ópticos baseiam-se na observação da interacção da luz com o objecto em análise e, em geral, não apresentam efeitos adversos em materiais biológicos. Estes sistemas requerem uma rotação de 360° do objecto em análise, ou múltiplas câmaras

(a) (b)

Fig. 2.3: *Scanners* 3D e equipamento de TC. (a) *Scanners* 3D: Mensi S25 (à esquerda), Cyrax 2500 (à direita), GOM ATOS II (em baixo) [13], (b) Equipamento de TC (adaptado de [14]).

posicionadas em torno do objecto, de forma a capturarem toda a sua superfície [5]. Os *scanners* ópticos podem ser activos ou passivos. Enquanto que os *scanners* ópticos activos necessitam de iluminação controlada, os *scanners* ópticos passivos dependem somente da luz ambiente.

- ***Scanners* ópticos passivos**

  Os *scanners* passivos não se servem da projecção de luz sobre a superfície do objecto em análise, dependendo somente da luz ambiente. São, na sua maioria, sistemas baseados em estereoscopia, utilizando duas câmaras para capturarem pontos de vista diferentes de um objecto [10, 15]. Através da correspondência de um mesmo ponto em imagens diferentes, e por triangulação, é possível traçar uma linha tridimensional desde o centro de projecção de cada câmara até ao ponto 3D (figura 2.4). A intersecção destas linhas é então utilizada para recuperar a profundidade do ponto no espaço, sendo previamente calibradas as câmaras, e consideradas as propriedades de cada uma [16].

  O estabelecimento de uma correspondência nos vários pontos de vista do objecto é um grande desafio, sendo desenvolvidos vários métodos para este fim [10]. A estereoscopia constitui um dos tópicos mais antigos e um dos mais investigados na comunidade da visão por computador [18].

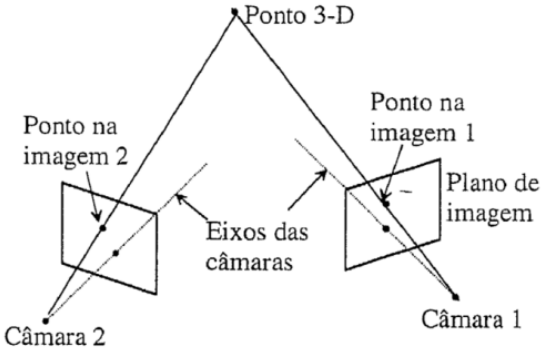


Fig. 2.4: Profundidade de um ponto por triangulação [17].

- ***Scanners* ópticos activos**

  A ideia base dos *scanners* ópticos activos é projectar um padrão ou um feixe de luz controlada na superfície do objecto [19]. Para tal, procede-se geralmente à substituição de uma das câmaras num sistema estereoscópico passivo, por uma fonte de luz (emissor de feixe de luz laser, ou projector de vídeo). Os *scanners* ópticos activos são mais sensíveis às propriedades das superfícies dos objectos do que os passivos. Objectos muito reflexivos ou translúcidos geralmente requerem tratamento e realização de várias medições adicionais para evitar erros no processo de digitalização [10]. As tecnologias de *scanners* ópticos mais utilizadas na construção de modelos 3D humanos baseiam-se na emissão de um feixe laser ou na projecção de padrões de luz estruturada. Estes métodos são activos, baseiam-se no princípio da triangulação e podem fornecer milhões de pontos, frequentemente com a informação de cor correspondente [20]. É importante referir que os *scanners* a laser e os *scanners* de luz estruturada são pouco eficazes na captura de cenas dinâmicas, e devido à separação entre o emissor e o receptor de luz, algumas regiões oclusas não são recuperadas. Além destes sistemas, existem os *scanners* 3D ópticos baseados em tempo de voo (do inglês, *time-of-flight*). Estes *scanners* determinam a posição de determinado ponto, considerando o tempo que um feixe de luz demora a percorrer a distância entre o emissor de luz e o objecto em análise [21]. Estes *scanners* são capazes de operar a longas distâncias, sendo aptos para digitalizações de edifícios, por exemplo. No entanto, devido à alta velocidade da luz, cronometrar o seu tempo de ida e volta é difícil [22].

2.2 *Scanners* 3D a Laser

Os *scanners* 3D a laser fazem parte dos sistemas ópticos activos, uma vez que projectam um feixe laser sobre o objecto a analisar. Em vez de duas câmaras como no caso dos sistemas estereoscópicos passivos, utiliza-se um emissor de feixe laser e uma câmara. Estes sistemas baseiam-se no princípio da triangulação. Na configuração mais comum, o emissor projecta um plano de luz laser na superfície do objecto em análise (*slit scanner*). O objecto distorce o plano emitido, e o contorno resultante, que corresponde à forma do objecto, é detectado pela câmara, como ilustra a figura 2.5.

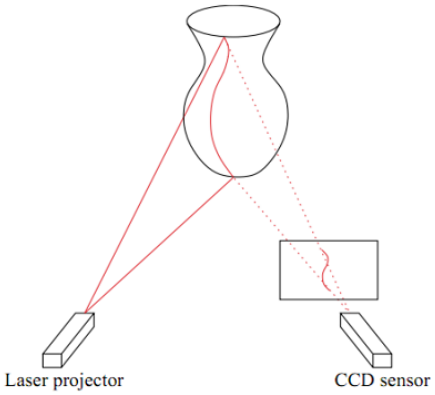


Fig. 2.5: Esquema de um *scanner* 3D a laser [23].

Ao projectar-se um simples plano de luz sobre o objecto, não se consegue obter informação sobre a sua estrutura completa. Por este motivo é necessário transladar o plano de luz, fazendo o varrimento do objecto. Este factor constitui uma das desvantagens deste método. A precisão do *scanner* é afectada pela precisão do sistema de posicionamento do feixe laser. Devido à sua robustez e à sua simplicidade, este tipo de *scanner* é muito popular. No entanto, estes *scanners* são difíceis de usar quando estão presentes objectos em movimento, sendo limitados à reconstrução de objectos estáticos [10].

2.3 *Scanners* 3D de Luz Estruturada

Tal como os *scanners* 3D a laser, os *scanners* 3D baseados na projecção de luz estruturada fazem parte dos sistemas ópticos activos. Em vez de projectarem somente um plano de luz sobre o objecto, os *scanners* 3D de luz estruturada projectam

um padrão, ou uma sequência de padrões codificados de luz na superfície do objecto. Desta forma, é eliminada a necessidade de varrimento do objecto, tornando o processo mais rápido e eficaz, o que constitui uma vantagem. Por outro lado, é necessário referir que o estabelecimento de uma correspondência entre o emissor dos padrões de luz (projector de vídeo) e o receptor (câmara) pode tornar-se mais difícil, mediante o tipo de padrões projectados. A figura 2.6 apresenta um esquema deste tipo de *scanners* 3D.

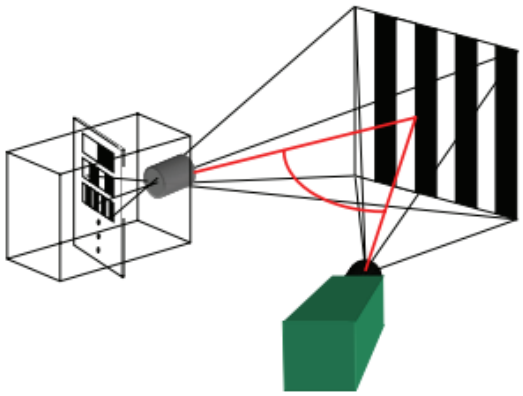


Fig. 2.6: Esquema de um *scanner* 3D de luz estruturada. Adaptado de [24].

Os padrões podem ser codificados espacialmente (num único *frame*), temporalmente (em múltiplos *frames*), ou numa combinação dos dois tipos de codificação. A codificação espacial consiste na projecção de um único padrão para a aquisição da estrutura do objecto, permitindo a sua aplicação em cenas dinâmicas. Por outro lado, a codificação temporal consiste na projecção de uma sequência de padrões e beneficia de redundância, reduzindo artefactos de reconstrução da estrutura do objecto [10]. Actualmente existe uma grande diversidade de técnicas para a codificação dos padrões projectados, sendo possível verificar uma comparação entre as técnicas mais utilizadas em [25, 26, 27].

2.3.1 Codificação Binária dos Padrões de Luz Estruturada

Uma das principais técnicas de codificação, inicialmente proposta por Posdamer e Altschuler em 1982 [28], consiste na projecção de uma sequência de padrões de luz com dois níveis de iluminação. Cada padrão corresponde a um conjunto de planos de luz codificados como 1 ou 0, mediante sejam iluminados ou não iluminados [27]. A determinação da posição de cada ponto na cena é feita por triangulação, pela intersecção do plano iluminado/não iluminado que passa por esse ponto, com a

linha de visão do ponto determinada pela imagem capturada pela câmara. Para tal, é necessário conhecer as equações do plano e da linha de visão (conseguidas através da calibração geométrica do projector e da câmara), e determinar a correspondência entre os planos projectados e os planos visíveis na imagem capturada pela câmara. A correspondência obtém-se a partir da atribuição de um código único a cada plano de luz ao longo da sequência de padrões de luz. Cada padrão de luz é constituído pelos planos de bits da representação binária dos códigos dos planos [17]. Para a codificação de 2^m planos de luz iluminados/não iluminados, é necessária a projecção de uma sequência de m padrões de luz. Por exemplo, para a codificação de oito planos de luz, será necessário projectar $\log_2 8 = 3$ padrões distintos, como se pode ver na figura 2.7.

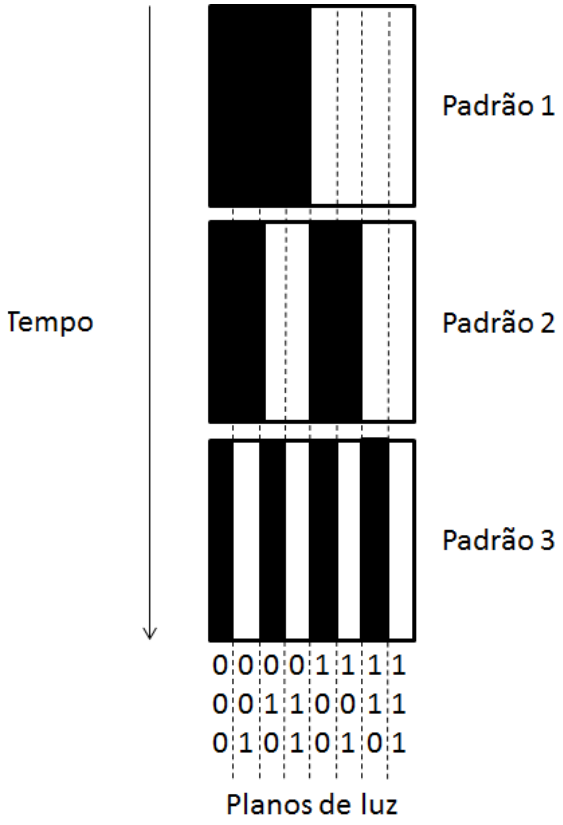


Fig. 2.7: Projecção de padrões de luz e codificação dos planos iluminados/não iluminados.

Cada plano de luz é codificado pela sequência de valores de luminosidade (0 e 1), correspondentes ao seu valor em cada um dos m padrões projectados, sendo o primeiro padrão aquele que contém o bit mais significativo. Entre projecções consecutivas, o número de planos projectados aumenta por um factor de 2, e o número máximo de planos que pode ser projectado corresponde à resolução máxima

do dispositivo de projecção.

2.3.2 Utilização de Código Gray

Os padrões projectados na técnica descrita em [28] baseiam-se em código binário simples. No entanto, em 1984 foi proposto, por Inokuchi et al. [29], a utilização de código Gray para tornar mais preciso o estabelecimento de correspondências entre os planos projectados e os planos visíveis na câmara. A grande vantagem do código Gray relativamente ao código binário simples, reside no facto de o código para dois planos vizinhos nos padrões projectados diferenciar-se em apenas um bit. A diferença entre os dois tipos de codificação pode ser verificada na figura 2.8.

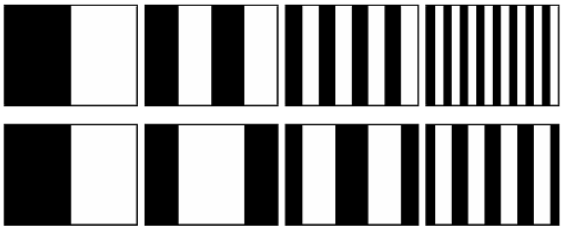


Fig. 2.8: Sequência de padrões de luz estruturada, correspondendo aos quatro primeiros planos de bits de uma sequência binária simples (em cima, da esquerda para a direita), e de uma sequência de código Gray (em baixo, da esquerda para a direita) [10].

Como resultado, a sequência de padrões de luz baseada em código Gray tende a ser mais robusta do que uma codificação binária simples [10].

2.3.3 Localização e Identificação dos Planos de Luz

A codificação dos planos de luz é feita através de código binário simples ou código Gray, e a determinação da posição de um ponto na cena é obtida por triangulação, através da intersecção do plano de luz com a recta que une o ponto ao centro de projecção da câmara. Para isso, é necessário identificar, em cada uma das m imagens obtidas ao longo da sequência de projecção de m padrões, se determinado pixel está iluminado pelo projector. Se estiver, o bit correspondente é definido como 1, caso contrário é definido como 0. Os planos projectados são identificados através da descodificação da sequência de bits para cada pixel da imagem captada pela câmara. Para o sistema identificar e localizar os vários planos de luz, distinguindo um plano iluminado de um plano não iluminado e do meio, procede da seguinte forma: além dos padrões mencionados, são projectados dois padrões extra, correspondendo a

um *frame* totalmente branco e a um *frame* totalmente preto. Assim, o total de projecções corresponde a $[log_2 m] + 2$ padrões. A intensidade média destes dois últimos *frames* pode ser utilizada para definir um *threshold* de intensidade de luz. Desta forma, para determinar se certo pixel está iluminado por um plano de luz, compara-se a intensidade desse pixel com o *threshold* definido. Se o valor da sua intensidade luminosa for superior ao valor de *threshold*, então corresponde a um plano iluminado e o valor passa a 1, caso contrário, passa a 0. No entanto, certos pontos na superfície do objecto podem ser indirectamente iluminados, recebendo luz dispersada de pontos iluminados pelo projector. Desta forma, um ponto não iluminado pode ser erradamente identificado como sendo iluminado, originando um erro de determinação de um bit, o que produzirá erros na determinação da posição do ponto no espaço. Uma solução para este problema consiste em projectar cada plano de bits e o seu inverso, tornando o processo de descodificação menos sensível à luz espalhada. Assim, um bit é colocado a 1 se o plano de bits projectado apresentar um nível de iluminação superior ao valor do threshold, e o seu inverso apresentar um nível de iluminação inferior ao valor do threshold [10].

2.4 Tomografia Computorizada

A tomografia computorizada (TC), também chamada de tomografia axial computorizada (TAC), foi introduzida na década de 70 por Hounsfield e Conrmack, o que lhes valeu um prémio Nobel em 1979 [2]. As primeiras imagens de TC foram produzidas no hospital Atkinson Morley em Londres em 1972, e ofereceram provas convincentes da efectividade do método desenvolvido, ao permitirem detectar um tumor cístico no lobo frontal de um paciente. A TC foi então bem recebida pela comunidade médica e é muitas vezes referida como a invenção mais importante na radiologia desde a descoberda dos raios-X. A boa aceitação da TC fez com que o número de *scanners* de TC instalados aumentasse de 60 em 1974, para mais de 10000 em 1980 [30]. Actualmente, estima-se que são feitos mais de 62 milhões de exames de TC por ano só nos Estados Unidos da América, dos quais 4 milhões de exames são realizados em crianças [15]. Para produzir raios-X, a TC utiliza a mesma técnica da radiografia. No entanto, para obter imagens axiais de secções do corpo em análise, o tubo emissor roda ao longo do eixo longitudinal (z) do corpo, transmitindo radiação a partir de várias posições angulares. Os raios-X são atenuados de acordo com a densidade radiográfica dos tecidos e atingem o sistema detector atrás do corpo em análise, contribuindo para o perfil de atenuação de um determinado

ângulo do tubo. Os vários perfis medidos durante uma rotação são utilizados pelo computador para calcular um mapa de densidade do corte analisado, com valores discretos de densidade radiográfica para todos os elementos da imagem (pixels). Na figura 2.9 é possível verificar um conjunto de imagens obtidas por TC.

(a) (b) (c)

Fig. 2.9: Imagens axiais obtidas por tomografia computorizada, correspondendo a diferentes regiões do corpo humano: (a) Cabeça, (b) tórax e (c) abdómen.

A radiodensidade é expressa em unidades Hounsfield (HU, do inglês *Hounsfield Units*). Na tabela 2.1 é possível verificar valores de referência para a radiodensidade de diferentes elementos, em condições de pressão e temperatura normais (PTN). Estes valores de referência foram escolhidos uma vez que são universais e adequados para a principal aplicação da TC: visualização da anatomia interna humana [2].

Tab. 2.1: Valores de referência de radiodensidade.

Tecido	Radiodensidade (HU)
Água	0
Ar	-1000
Gordura	-120
Músculo	+40
Osso	+1000
Metal	> +1000

Ao longo dos anos, os *scanners* de TC têm apresentado uma grande evolução, passando de sistemas capazes de adquirir somente imagens da cabeça, para sistemas capazes de analisar e obter informação volumétrica de todo o corpo humano. Os primeiros *scanners* de TC demoravam cerca de 25 minutos para obter imagens da cabeça de um paciente, e o emissor e o detector de raios-X descreviam movimentos de rotação e de translação em torno do paciente. O paciente era movido

discretamente segundo o eixo perdendicular à linha que unia o emissor e o detector de raios-X. Actualmente, os *scanners* de TC apresentam um emissor e uma matriz de detectores posicionados numa *gantry* que descreve um movimento helicoidal em torno do paciente. O paciente descreve um movimento contínuo, a uma velocidade constante, axialmente através do *scanner* de TC. Assim, os *scanners* de TC podem adquirir até 128 cortes numa única rotação do tubo emissor, e os exames demoram poucos segundos. Uma revisão completa da evolução da TC pode ser consultada em [30]. Uma vantagem evidente da TC é o facto de obter informação do interior do corpo humano e também de objectos. Por outro lado, devido à emissão de raios-X, a exposição dos operadores e dos pacientes a este tipo de radiação ionizante constitui uma desvantagem. A redução do risco de exposição desnecessária requer o treino dos operadores e a combinação de várias medidas de segurança [15].

2.5 Aplicações dos *Scanners* 3D

Os *scanners* 3D são ferramentas de base no processo de reconstrução de superfícies nas mais variadas áreas. Desde a Medicina à produção cinematográfica, passando pelos vários tipos de indústria, é possível verificar aplicações destes sistemas. Nesta secção são apresentados exemplos de aplicações e de trabalhos recentes em diversas áreas.

2.5.1 Medicina Forense

Os *scanners* 3D têm tido uma utilização crescente na Medicina Forense, sendo utilizados na aquisição de informação 3D de vítimas, de objectos, e até por vezes da própria cena do crime. Os objectivos da aquisição de informação 3D são vários, sendo possível referir a documentação de traumas em vítimas e de potenciais armas do crime, preservação de estruturas, reconstruções 3D de vítimas para a sua identificação, e até a digitalização de pegadas para se descobrir o sapato utilizado por um criminoso [31, 3, 4].

Reconstrução facial

Um trabalho na área da Medicina Forense, desenvolvido por P. Vanezis *et al*, procura facilitar a identificação de sujeitos em decomposição ou desfigurados, recorrendo a *scanners* 3D a laser e a tecnologias de reconstrução 3D [31]. A descoberta de restos humanos já em decomposição ou desfigurados, deixa os investigadores com pouca

informação para proceder à sua identificação. Nestes casos, o investigador fica limitado a aspectos gerais de identificação tais como a idade, a raça, o sexo, a estatura e constituição. Se, após terem sido explorados os meios convecionais, não for possível identificar os restos humanos, surge a possibilidade de se reconstruir a face a partir de um crânio, se este estiver disponível. O objectivo da reconstrução facial não passa pela obtenção de uma face exactamente igual à do sujeito *ante-mortem*, mas sim pela reconstrução de um modelo que apresente características do sujeito e facilite a sua identificação. O primeiro passo do trabalho consiste numa avaliação antropológica do crânio para definição do sexo, da idade, da raça e, se possível, da estatura. Seguidamente, o crânio é preparado para a aquisição 3D. Para tal, todos os orifícios são tapados com algodão ou com uma sustância similar, para que o feixe laser projectado não passe através do crânio. Passa-se à aquisição da estrutura 3D do crânio utilizando um *scanner* 3D a laser e, através de um conjunto de algoritmos computacionais, é gerada a superfície 3D correspondente. É então colocado um conjunto de marcadores em posições específicas da superfície 3D (figura 2.10). A cada marcador é atribuída uma numeração, uma designação específica, e é associado um tipo de espessura que permite definir o tipo de tecido na região do marcador e um tipo antropológico (conjunto de características físicas). A fase seguinte consiste na selecção de um modelo facial de uma base de dados de faces previamente digitalizadas no sistema. É seleccionado o modelo que apresente características mais similares às do crânio e que seja correspondente antropologicamente. Os marcadores são então colocados no modelo seleccionado, nas mesmas posições espaciais dos marcadores no crânio adquirido pelo *scanner* 3D (figura 2.10). É feito um mapeamento entre os marcadores no crânio e os marcadores no modelo da face e é calculada uma transformação geométrica para se ajustar a face ao crânio (figura 2.10).

Em termos gerais, a aplicação de *scanners* 3D em reconstruções faciais apresenta algumas vantagens sobre os métodos tradicionais, nomeadamente:

- Torna desnecessária a criação de um molde do crânio;
- O método de aquisição de informação 3D do crânio é não-invasivo e não-destrutivo;
- O crânio pode ser armazenado com segurança e a sua integridade é totalmente preservada.

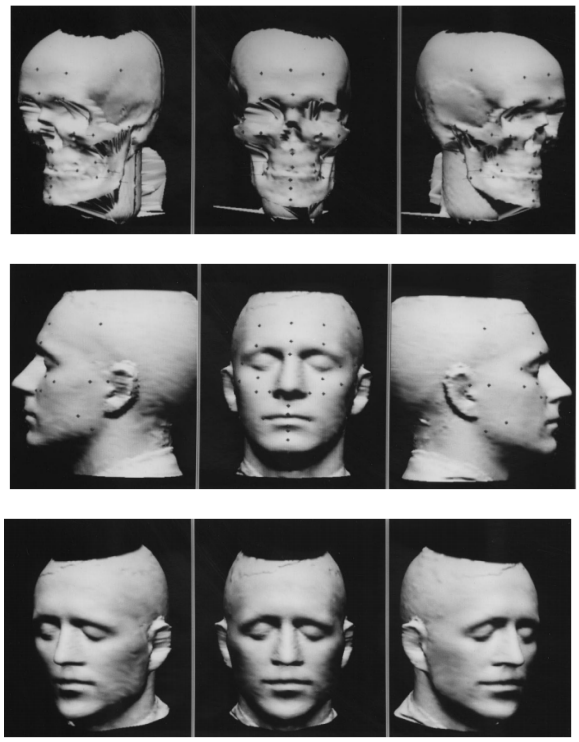


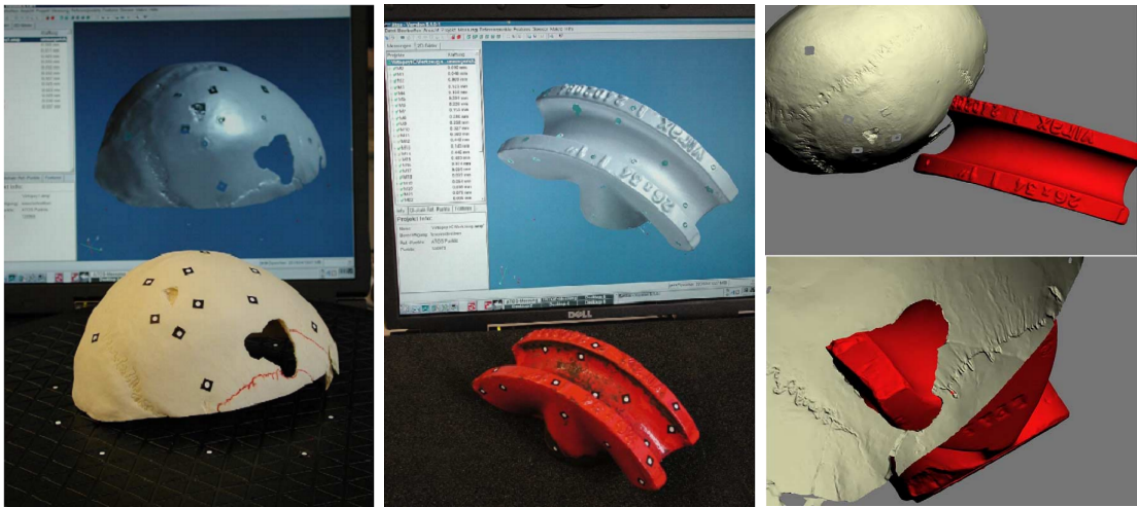
Fig. 2.10: 1ª Linha: Perspecitvas da superfície 3D do crânio com os marcadores em posição. 2ª Linha: Perspectivas do modelo facial seleccionado, com os marcadores em posição. 3ª Linha: Perspectivas da face obtida [31].

Identificação de Armas do Crime

Os *scanners* 3D também têm sido utilizados para adquirir informação 3D de traumas em vítimas e de possíveis armas de crimes [3]. Além da documentação, os objectivos destas aquisições passam pela descoberta da arma ou do objecto utilizado para provocar dano na vítima. Após serem criadas as respecitvas superfícies 3D, pode ser verificada a compatibilidade entre a forma da lesão na vítima e a forma de determinado objecto, sendo possível concluir se esse objecto foi ou não a arma do crime. Na figura 2.11, é possível verificar o estabelecimento de uma correspondência entre um trauma num crânio e a forma de um objecto.

Extracção de informação a partir de objectos

Outra aplicação dos *scanners* 3D na Medicina Forense, consiste na aquisição de informação 3D de objectos encontrados numa cena do crime, mais propriamente de pastilhas elásticas. É possível extrair das pastilhas elásticas informações como ADN, antigénios do grupo sanguíneo e impressões de dentes. No entanto, alguns testes para



(a) (b) (c)

Fig. 2.11: Correspondência entre um trauma e um objecto através da análise de superfícies 3D. (a) Superfície 3D do crânio, (b) Superfície do objecto, (c) Correspondência entre o trauma e o objecto [3].

extrair informações do ADN podem deformar as pastilhas, impossibilitando assim uma modelação odontológica precisa. Assim, é importante referir o trabalho de R. Siderits *et al* [4]. Este trabalho apresenta um método para aquisição de informação 3D de uma pastilha elástica, usando um *scanner* 3D a laser, e para a criação de modelos sólidos reais com auxílio de software CAD (*Computer Aided Design*). O objectivo final deste trabalho é a preservação da estrutura dos dentes, permitindo a realização segura de testes para extração de ADN das pastilhas elásticas. Na figura 2.12, são apresentados modelos em plástico de pastilhas elásticas e de ossos do hamato (pulso), obtidos com esse trabalho.

2.5.2 Área da Saúde

Cirurgia plástica

A partir da reconstrução 3D da região do paciente que será submetida a uma intervenção cirúrgica, é possível realizar uma modelação preliminar, fornecendo ao paciente uma imagem completa das mudanças previstas após a intervenção. Isto também permite aos médicos apresentar um argumento convincente a favor ou contra uma determinada intervenção cirúrgica. Os *scanners* 3D podem fornecer aos médicos três modelos: pré-operatório, previsão do resultado e pós-operatório, o que permite uma avaliação precisa do sucesso da operação [32].

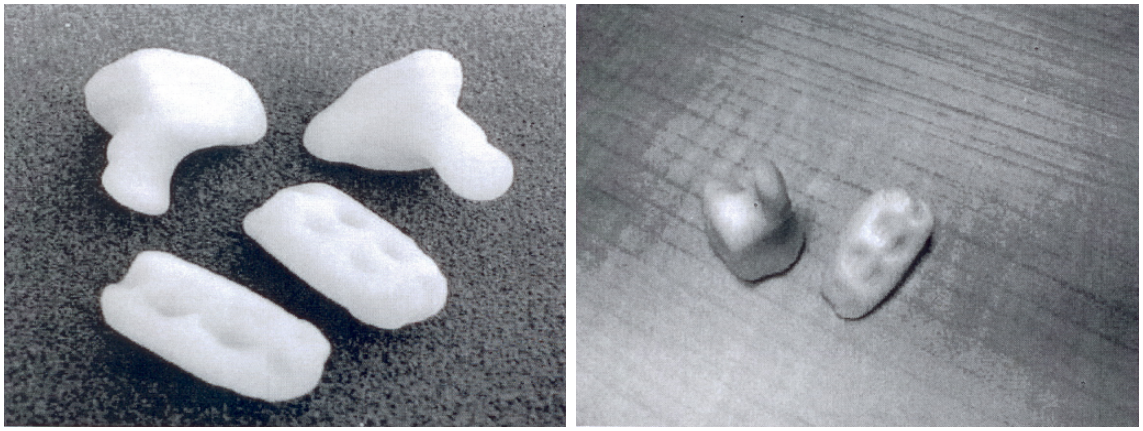


Fig. 2.12: Modelos de plástico. Pastilha elástica e ossos do hamato [4].

Ortodontia

As aplicações dos *scanners* 3D em Ortodontia incluem as avaliações pré e pós-ortodontica de relações dentoesqueléticas e estética facial, o planeamento tridimensional de tratamentos, bem como a simulação de tratamentos. É possível arquivar os registos faciais, esqueléticos e dentários 3D no planeamento de tratamentos. Actividades de pesquisa e fins médico-legais também estão entre os benefícios do uso de superfícies 3D em ortodontia. Uma vantagem evidente é que, com a captura da forma 3D dos dentes de um paciente, pode-se subsituir o desagradável uso de pasta para moldes. Após a aquisição de informação tridimensional das estruturas de interesse, é utilizado software CAD/CAM para desenhar e produzir a respectiva prótese ou implante dentário [33]. A figura 2.13 apresenta a previsão de uma deslocação mandibular após cirurgia a partir de uma reconstrução 3D facial.

Ortopedia e cadeiras de rodas

Os *scanners* 3D são usados em Ortopedia na detecção de doenças como escoliose, cifose, lordose e espondilolistese rotacional num estado inicial de evolução. A aquisição de informação tridimensional das costas de um paciente é um procedimento rápido e sem riscos para a saúde, constituíndo um método de avaliação seguro e que se pode realizar as vezes necessárias. Desta forma, a utilização de um *scanner* 3D para medir a curvatura da coluna de um paciente representa uma ajuda no diagnóstico da evolução de uma das doenças supracitadas. Por outro lado, a reconstrução 3D das costas de um paciente facilita o processo de criação de uma cadeira de rodas que satisfaça por completo as necessidades de um paciente especial [5].

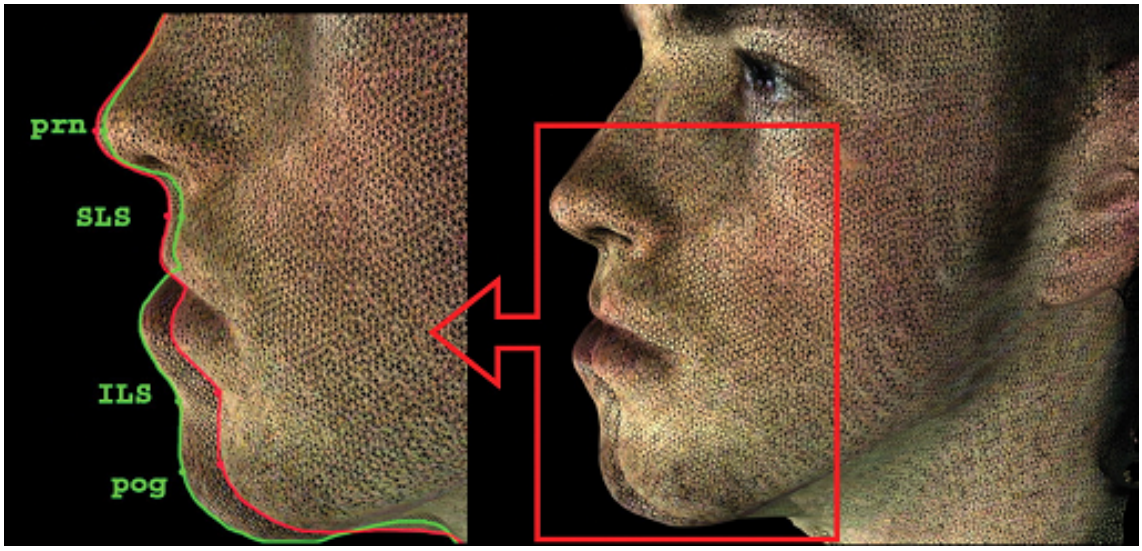


Fig. 2.13: Previsão do pós-operatório a partir de uma reconstrução 3D [33].

Tratamento pós-traumático

Uma vez que o tratamento de traumas graves pode ser tão ou mais doloroso que o trauma em si, a utilização de *scanners* 3D pode constituir um grande alívio para o paciente. Os *scanners* podem fornecer uma superfícies 3D precisa da face de um paciente queimado, por exemplo, o que vai permitir aos médicos a criação de uma máscara personalizada para agilizar o processo de cura sem necessidade de moldes de gesso, ou a necessidade de contacto com uma área antigida do corpo [5].

Nutrição

Através da reconstrução 3D da superfície do corpo de um paciente, os dietistas podem mostrar aos seus pacientes a evolução resultante do regime que estão a utilizar, e também ir ajustando o regime. Os pacientes que recebem confirmação visual da efectividade do tratamento a que se estão a submeter sentem-se mais incentivados para continuar com o tratamento e para atingir o fim pretendido [5].

2.5.3 Outras Áreas de aplicação

Arte e Arqueologia

Os *scanners* 3D são utilizados na área da Arte e Arqueologia para a documentação de património cultural. Este é o caso do trabalho de D. Akca *et al* [24], cujo objectivo passa pela documentação 3D de uma estátua, utilizando um *scanner* 3D

óptico baseado na projecção de luz estruturada. A estátua encontra-se partida em duas partes, sendo que a parte adquirida pelo *scanner* 3D corresponde aos membros inferiores de Heracles, filho de Zeus, e encontra-se no Museu de Antalya na Turquia. A figura 2.14 apresenta a estátua a partir da qual foi obtida informação 3D, e a respectiva superfície 3D originada.

Fig. 2.14: Estátua de Heracles e respectivo modelo 3D [24].

Os objectivos da reconstrução de superfícies 3D a partir de património cultural passam pela sua catalogação, pelo planeamento de restaurações, pela reprodução de peças partindo de modelos CAD e pela monitorização do seu estado ao longo do tempo. Outro trabalho relevante nesta área é o famoso *Digital Michelangelo project* de M. Levoy *et al* [34], no qual é construído um *scanner* 3D baseado em tempo de vôo, projectando um feixe laser. O objectivo deste trabalho consite em contruir modelos 3D de estátuas de Miguel Ângelo, de interiores de edifícios históricos, e do *Forma Urbis Romae*, um mapa gigante em mármore da Roma Antiga. A figura 2.15 apresenta a face da estátua de David por Miguel Ângelo, e a respectiva superfície 3D reconstruída.

Indústria e Controlo da Qualidade

Os *scanners* 3D são utilizados na indústria para controlo da Qualidade, engenharia inversa e prototipagem. No caso da engenharia inversa, os *scanners* 3D são utilizados na aquisição de informação tridimensional de objectos físicos para permitir reconstruir superfícies 3D. O objectivo final é a diminuição do tempo utilizado no

(a) (b)

Fig. 2.15: Estátua de David e respectiva superfície 3D. (a) Estátua real de David, (b) Superfície 3D da estátua obtida com um *scanner* a laser [34].

processo desenho-mercado, através da combinação de tecnologias como CAD/CAM para processos de replicação. Por outro lado, como os dados geométricos dos componentes podem ser fácil e rapidamente medidos, é possível realizar comparações rápidas entre peças fabricadas e um desenho CAD original, o que constitui um papel fundamental no controlo de Qualidade do produto. Por fim, os *scanners* 3D podem ser usados para digitalizar protótipos, permitindo assim a criação de superfícies 3D virtuais que podem ser alterados digitalmente e, em combinação com uma impressora 3D, podem ser criados fisicamente [35].

Indústria de Vestuário e Moda

Os *scanners* 3D têm apresentado uma utilização crescente na indústria do vestuário e moda. As suas aplicações são várias, sendo possível enunciar a criação de sistemas para provação virtual de roupa e de equipamentos; a criação de *avatares* e de modelos para videojogos e filmes; a criação de objectos e peças de roupa à medida [36, 5]. A figura 2.16 apresenta alguns modelos 3D virtuais construídos a partir de informação 3D do corpo humano.

Como é possível verificar, existem inúmeras aplicações para os *scanners* 3D nas mais variadas áreas. Além das áreas de aplicação enumeradas anteriormente, é possível referir áreas como construção civil, por exemplo. Esta grande aplicabilidade

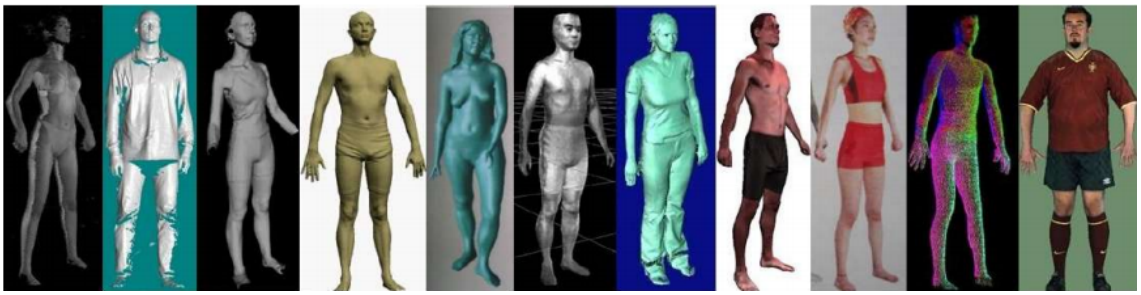


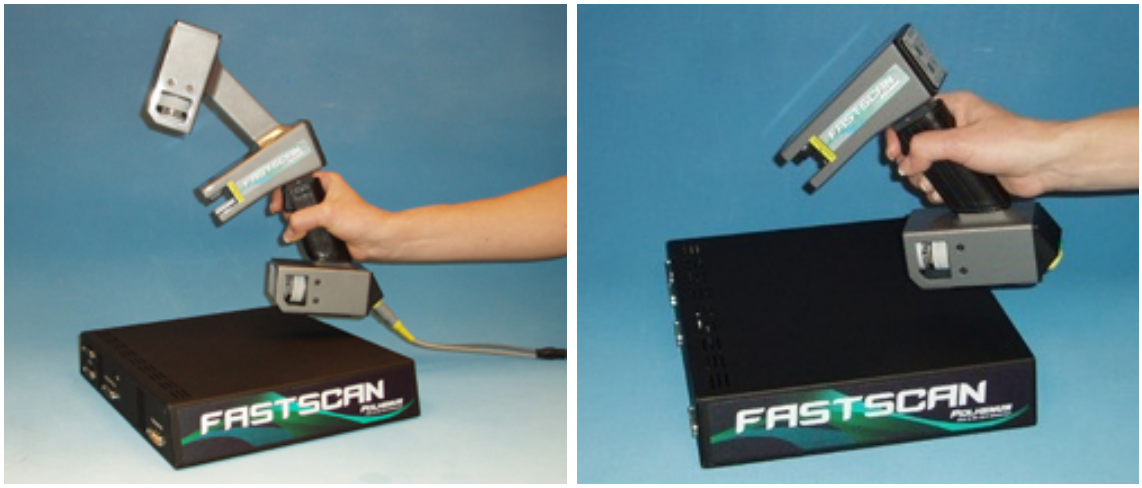
Fig. 2.16: Modelos 3D virtuais, construídos a partir de informação 3D do corpo humano [36].

dos *scanners* 3D resulta da sua facilidade de construção e de utilização, bem como das suas vantagens específicas em cada caso, quando comparados com outros meios.

2.6 *Scanners* Comerciais

Actualmente existem várias empresas fabricantes de *scanners* 3D para as áreas descritas na secção anterior (secção 2.5). Uma empresa de destaque é a *Polhemus*, situada nos Estados Unidos da América, fabricante do *scanner* *FastSCAN*™[37]. Este *scanner* óptico a laser adquire instantaneamente pontos da superfície em análise e permite a visualização dos resultados em tempo real, sendo aplicado em ortopedia, tratamentos crânio-faciais, e em cirurgia plástica, na avaliação pré e pós-operatória. O *scanner* *FastSCAN*™está disponível em duas versões, uma com duas câmaras (*Scorpion*) e outra com uma câmara (*Cobra*), visíveis na figura 2.17 (a) e (b) respectivamente. Outra empresa a referenciar é a Alemã *GOM - Gesellschaft für Optische Messtechnik*. Esta empresa produz *scanners* 3D ópticos direccionados para o desenvolvimento de produto, controlo de qualidade e testes de materiais. O *scanner* 3D mais famoso desenvolvido por esta empresa é o *ATOS Compact* [38]. Este *scanner* óptico é compacto e rápido, sendo usado na análise de protótipos, componentes industriais, ferramentas, entre outros. Na figura 2.17 (c) é possível observar o *scanner* *ATOS*.

Por fim, uma empresa relevante a nível internacional, que também produz *scanners* 3D, sistemas de medição CAD e *software* de processamento 3D é a empresa *FARO*. Os sistemas produzidos pela *FARO* são utilizados em várias áreas como os sectores aeroespacial, automóvel, do mobiliário e da saúde. As aplicações são várias, passando pelo alinhamento de peças em linhas de montagem, verificação das dimensões de peças produzidas, engenharia inversa, e documentação de edifícios.



(a) (b)

(c) (d)

Fig. 2.17: *Scanners* 3D comerciais. (a) *FastSCAN*$^{\text{TM}}$*Scorpion*, (b) *FastSCAN*$^{\text{TM}}$*Cobra*, (c) *ATOS Compact*, (d) *Focus 3D*.

Um *scanner* 3D relevante, produzido pela *FARO* é o *Focus 3D* [39] (ver figura 2.17 (d)). Este *scanner* a laser de longo alcance é utilizado em Medicina Forense, na reconstrução e documentação de cenas do crime ou de acidentes. Os benefícios da sua utilização são vários, podendo enumerar o arquivo permanente e detalhado de cenas do crime e de acidentes, simulação 3D dos eventos ocorridos, e desenvolvimento de planos de emergência e de medidas de segurança através de representações virtuais. Em Portugal existe uma empresa chamada *Superfície*, que não produz *scanners* 3D, mas que recorre a estes sistemas para prestar serviços de levantamento de informação

3D nas áreas da Arqueologia, Arquitectura, conservação e restauro de património, e registo gráfico sub-aquático. Alguns dos trabalhos já realizados consistiram na digitalização do túmulo da Rainha Santa Isabel, utilizando um *scanner* de luz estruturada, e a digitalização do túmulo de Pedro o Grande, utilizando um *scanner* a laser. Trabalhos realizados pela empresa *Superfície* podem ser consultados em [40].

2.7 Considerações Gerais

Actualmente existem vários *scanners* 3D baseados nos sistemas abordados nesta secção, levando ao desenvolvimento de trabalhos para comparação e *benchmark* dos vários sistemas, como se pode verificar em [41]. Dependendo da utilização prentedida, está disponível um grande conjunto de sistemas, devendo a escolha recair sobre a tecnologia mais adequada. Se o objectivo for analisar externa e internamente um objecto, um *scanner* de TC é uma solução adequada já que permite adquirir com elevado detalhe e precisão a forma das estruturas em estudo. No entanto, além do *scanner* de TC ser um equipamento caro, emite radiação ionizante. Uma alternativa para adquirir informação 3D de estruturas, quando só se pretende informação da superfície externa, consiste na utilização de *scanners* 3D não baseados na emissão de radiação ionizante. *Scanners* 3D por contacto e *scanners* 3D ópticos reflexivos são boas alternativas à TC, constituindo meios mais económicos e menos perigosos para a aquisição de informação 3D. Em termos gerais, tanto os *scanners* de contacto como os *scanners* sem contacto apresentam as suas vantagens e desvantagens, sendo possível referir algumas delas. Os *scanners* de contacto, devido à interacção com os objectos a analisar, têm a desvantagem de poderem comprometer a sua estrutura física. Assim sendo, este tipo de *scanner* seria desaconselhado por exemplo, para a aquisição de informação 3D de evidências numa cena do crime ou de peças de arte sensíveis. Outro ponto negativo a adicionar é o facto dos *scanners* de contacto serem significativamente mais lentos do que os *scanners* sem contacto. No entanto, o facto de os *scanners* por contacto não serem prejudicados por superfícies transparentes ou reflexivas, como é o caso dos *scanners* ópticos, e o facto de serem bastante precisos, constituem grandes vantagens. Já os *scanners* ópticos, têm a vantagem de não necessitar de contacto com os objectos. No entanto, encontram dificuldades ao analisarem objectos transparentes ou extremamente reflexivos. Existem alguns métodos para contornar os problemas associados à análise de objectos transparentes ou reflexivos, como é o caso de os cobrir com uma camada fina de pó, de forma a permitir que mais fotões sejam reflectidos para o *scanner*. Considerando objectos

que não possam ser sujeitos a tratamentos para facilitarem a análise por parte dos *scanners* ópticos, está encontrada uma limitação à sua utilização. Assim, um *scanner* deve ser seleccionado considerando todos os requisitos para a análise pretendida, bem como as suas características e limitações.

Capítulo 3

Triangulação Óptica

Os sistemas ópticos passivos e os sistemas ópticos activos adquirem informação 3D de forma semelhante. Em primeiro lugar, procede-se à calibração dos elementos que constituem o sistema. Estes elementos são câmaras, no caso dos sistemas puramente passivos, mas também emissores de luz (emissores laser, projectores de vídeo), no caso dos sistemas activos. É estabelecida uma correspondência entre câmaras, ou entre emissor-câmara, recorrendo-se a um método passivo ou a um método activo (secção 2.1.2). Sendo estabelecidas as correspondências, a posição dos pontos no espaço é obtida por triangulação. Neste capítulo será feita uma breve descrição do princípio da triangulação óptica, tendo por base o modelo *pinhole* para representação da geometria do sistema de aquisição.

3.1 Modelo *Pinhole*

O modelo utilizado para a representação da câmara e do projector é o modelo *pinhole* (figura 3.1). Segundo este modelo, uma imagem é formada numa câmara, através da projecção de pontos 3D da superfície de um objecto, no plano de imagem da câmara (projecção em perspectiva). No modelo adoptado, as coordenadas cartesianas são denotadas por (x, y, z) para pontos 3D físicos, expressos em unidades físicas (mm), e (u, v) para pixels 2D no plano da imagem, expressos em unidades de pixels (pix).

A geometria do projector é definida de forma semelhante, com a excepção de que, ao contrário do que acontece com a câmara, a luz *viaja* para fora do projector.

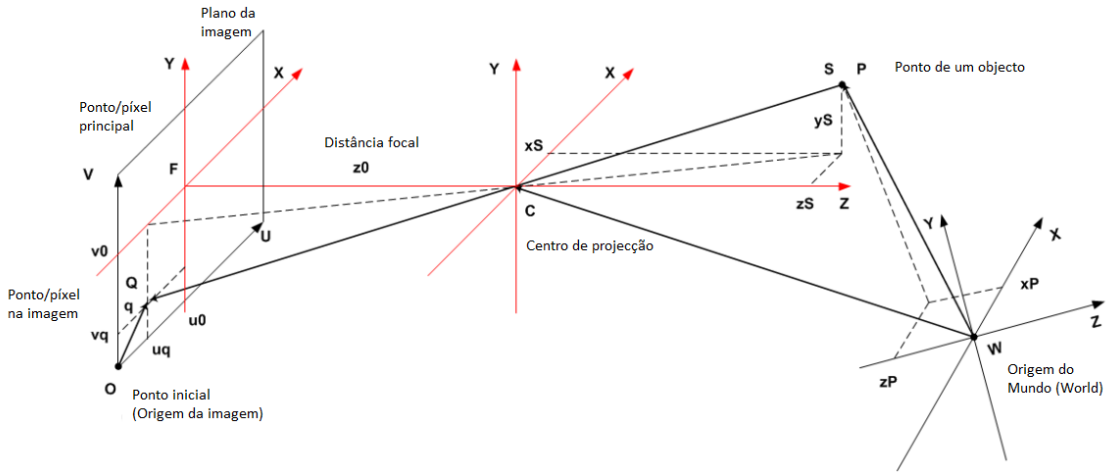


Fig. 3.1: Modelo *pinhole*.

3.1.1 Modelo Geométrico Adoptado

WXYZ - Sistema de coordenadas do mundo (mm)

As coordenadas (x, y, z) de pontos 3D físicos e distâncias são expressas em unidades físicas (mm). WXYZ é utilizado para descrever as coordenadas 3D de qualquer ponto P de um objecto, e o centro de projecção da câmara C (posição da câmara no mundo).

- $W = (0, 0, 0)^T$, origem do mundo
- $C^{(W)} = (-t_1, -t_2, -t_3)^T$, centro de projecção relativamente a W
- $P^{(W)} = (x_p, y_p, z_p)^T$, ponto dum objecto relativamente a W

CXYZ - Sistema de coordenadas da câmara (mm)

As coordenadas (x, y, z) de pontos 3D físicos e distâncias são expressas em unidades físicas (mm). A origem do sistema de coordenadas da câmara corresponde ao seu centro de projecção C. CXYZ é utilizado para descrever as coordenadas 3D da origem do mundo W, as coordenadas de um ponto S (a projectar no plano da imagem), e o ponto principal F (ponto de intersecção do eixo óptico com o plano da imagem, geralmente o centro da imagem). Todos os pontos projectados no plano da imagem passam pelo centro de projecção da câmara, C. Assumindo que o plano da imagem está localizado ao longo de CZ (eixo óptico) em $z = z_0$, e que o objecto está localizado em $z = z_S, z_S > 0$, a imagem correspondente é formada no plano

da imagem da câmara ($z = z_0, z_0 < 0$). Uma vez que $z_0 < 0$, a imagem aparece invertida no plano da imagem.

- $C = (0, 0, 0)^T$, origem da câmara
- $W^{(C)} = (t_1, t_2, t_3)^T$, origem do mundo
- $S^{(C)} = (x_S, y_S, z_S)^T$, ponto de um objecto

No plano da imagem, um ponto 3D Q projectado, resultante da projecção de um dado ponto 3D S, pode ser descrito em relação a C.

- $O^{(C)} = (x_0, y_0, z_0)^T$, origem da imagem (canto superior esquerdo)
- $G^{(C)} = (x_G, y_G, z_0)^T$, centro da imagem
- $F^{(C)} = (0, 0, z_0)^T$, ponto principal
- $Q^{(C)} = (x_Q, y_Q, z_0)^T$, ponto projectado

Uma imagem é definida em cada dimensão pelo seu tamanho (número de pixels), pelo espaçamento entre pixels e pelas dimensões físicas (em mm).

- N_X, N_Y, N_Z
- d_X, d_Y, d_Z
- $\Delta_X, \Delta_Y, \Delta_Z$

OUV - Sistema de coordenadas do plano da imagem (pix)

Pixels 2D no plano da imagem são expressos em unidades de pixels (pix), a partir do canto superior esquerdo O, a origem do plano da imagem.

- $O = (0, 0)^T$, pixel inicial (canto superior esquerdo)
- $G^{(O)} = (\frac{N_x}{2}, \frac{N_y}{2})^T$, pixel central
- $F^{(O)} = (u_0, v_0, z_0)^T$, pixel principal
- $q^{(O)} = (u_q, v_q)^T$, pixel correspondente à projecção de um ponto Q

3.1.2 Transformações de Coordenadas

1) Transformação rígida: $P^{(W)} \rightarrow S^{(C)}$

WXYZ (ponto de um objecto, mm) → CXYZ (ponto de um objecto, mm)
Um ponto de um objecto $P^{(W)}$ no sistema de coordenadas do mundo WXYZ é descrito pelo ponto correspondente $S^{(C)}$ relativamente ao sistema de coordenadas da câmara CXYZ. Estes dois vectores encontram-se relacionados por uma transformação rígida descrita por um vector de translação $\mathbf{T} \in \mathbb{R}^3$ e por uma matriz de rotação $\mathbf{R} \in \mathbb{R}^{3\times3}$. Os parâmetros ($\mathbf{R}$,$\mathbf{T}$) são designados *parâmetros extrínsecos* da câmara, e descrevem a localização do centro de projecção C, e a orientação da câmara relativamente ao sistema de coordenadas do mundo, tal que:

$$S^{(C)} = \mathbf{R}P^{(W)} + \mathbf{T} \tag{3.1}$$

$$\begin{bmatrix} x_S \\ y_S \\ z_S \end{bmatrix} = \begin{bmatrix} r_{11} & r_{21} & r_{31} \\ r_{12} & r_{22} & r_{32} \\ r_{13} & r_{23} & r_{33} \end{bmatrix} \begin{bmatrix} x_P \\ y_P \\ z_P \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} \tag{3.2}$$

Numa forma mais compacta:

$$S^{(C)} = \Big[\mathbf{R}|\mathbf{T}\Big] P^{(W)} \tag{3.3}$$

$$\begin{bmatrix} x_S \\ y_S \\ z_S \end{bmatrix} = \begin{bmatrix} r_{11} & r_{21} & r_{31} & t_1 \\ r_{12} & r_{22} & r_{32} & t_2 \\ r_{13} & r_{23} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} x_P \\ y_P \\ z_P \\ 1 \end{bmatrix} \tag{3.4}$$

A matriz $\Big[\mathbf{R}|\mathbf{T}\Big]$ é chamada de matriz dos *parâmetros extrínsecos* e é usada para descrever a posição e orientação da câmara em relação a uma determinada cena.

2) Modelo *pinhole* ideal: $S^{(C)} \rightarrow Q^{(C)}$

CXYZ (ponto de um objecto, mm) → CXYZ (ponto no plano da imagem, mm)
A relação entre as coordenadas CXYZ de um ponto 3D S, e a sua projecção Q no plano da imagem, $Q^{(C)} = (x_Q, y_Q, z_0)^T$ é descrita pelo modelo *pinhole* ideal. Em relação a C:

$$\lambda Q^{(C)} = S^{(C)}, \tag{3.5}$$

ou ainda,

$$\begin{cases} x_Q = Mx_S \\ y_Q = My_S \\ z_0 = Mz_S \end{cases}, \tag{3.6}$$

onde $M = 1/\lambda$ corresponde à ampliação da câmara, também dado por:

$$M = \frac{z_0}{z_S} \tag{3.7}$$

Se $z_0 < 0$, então $M < 0$ e a imagem aparece invertida no plano da imagem. Adicionalmente, a ampliação depende do ponto S, já que M está relacionado com a sua coordenada z_S.

3) Modelo *pinhole* geral: $S^{(C)} \rightarrow q^{(O)}$

CXYZ (ponto de um objecto, mm) → OUV (pixel na imagem, mm)
Uma vez que as distâncias no plano da imagem são medidas em unidades de pixels (pix), é necessário introduzir uma matriz $\mathbf{A} \in \mathbb{R}^{3\times3}$ que descreva a geometria interna da câmara (*parâmetros intrínsecos*), de modo a permitir transformar as coordenadas de um ponto $S^{(C)}$ segundo CXYZ, em coordenadas segundo OUV de um pixel $q^{(O)}$ na imagem, tal que:

$$\lambda q^{(O)} = \mathbf{A} S^{(C)}. \tag{3.8}$$

As coordenadas de um pixel $q_0 = (u_q, v_q)^T$ na imagem, podem ser descritas em pix como:

$$\lambda \begin{bmatrix} u_q \\ v_q \\ 1 \end{bmatrix} = \begin{bmatrix} -f_x & 0 & u_0 \\ 0 & -f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_S \\ y_S \\ z_S \end{bmatrix}, \tag{3.9}$$

$$\begin{cases} u_q = -\frac{f_x}{z_S} x_S + u_0 \\ v_q = -\frac{f_y}{z_S} y_S + v_0 \end{cases}. \tag{3.10}$$

A matriz $\mathbf{A} \in \mathbb{R}^{3\times3}$ é chamada matriz da câmara ou matriz dos *parâmetros intrínsecos*. Os parâmetros intrínsecos (u_0, v_0) definem o pixel principal F no sistema de coordenadas da imagem (OUV), isto é: $F^{(0)} = (u_0, v_0)^T$, e f_x, f_y ($f_x > 0, f_y > 0$)

são as distâncias focais expressas em unidades pix. Uma vez que a matriz **A** é independente do posicionamento da câmara e da cena observada, uma vez estimada, pode ser reaplicada.

4) Modelo *pinhole* completo: $P^{(W)} \rightarrow q^{(O)}$

WXYZ (ponto de um objecto, mm) → OUV (pixel na imagem, mm)
O modelo *pinhole* completo contempla os parâmetros intrínsecos e extrínsecos da câmara, permitindo determinar as coordenadas no plano da imagem OUV de um pixel $q^{(O)}$, correspondente à projecção de um dado ponto 3D $P^{(W)}$, dadas as suas coordenadas WXYZ:

$$\lambda q^{(O)} = \mathbf{A}(\mathbf{R}P^{(W)} + \mathbf{T}) \tag{3.11}$$

$$\lambda q^{(O)} = \mathbf{A}\left[\mathbf{R}|\mathbf{T}\right] P^{(W)} \tag{3.12}$$

$$\lambda \begin{bmatrix} u_q \\ v_q \\ 1 \end{bmatrix} = \begin{bmatrix} -f_x & 0 & u_0 \\ 0 & -f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{21} & r_{31} & t_1 \\ r_{12} & r_{22} & r_{32} & t_2 \\ r_{13} & r_{23} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} x_P \\ y_P \\ z_P \\ 1 \end{bmatrix} \tag{3.13}$$

3.2 O Princípio da Triangulação

Num *scanner* 3D óptico activo, uma linha emitida pelo projector origina um plano de luz (o único plano que contém a linha no plano da imagem e que passa pelo seu centro de projecção), e um ponto projectado origina um raio de luz (a única recta que contém o ponto no plano da imagem e que passa pelo centro de projecção). A intersecção de um raio de luz com o objecto em análise pode ser considerada como um único ponto, enquanto que a intersecção de um plano de luz com o objecto contém vários pontos. Um ponto no objecto iluminado pelo projector e visível pela câmara, define um raio da câmara. Conhecendo os *parâmetros intrínsecos* e *extrínsecos* do projector e da câmara, é possível determinar a profundidade dos pontos iluminados, ou seja, a distância a que os pontos iluminados se encontram da câmara. Para tal, são intersectados os planos ou raios de luz emitidos pelo projector, com os raios da câmara correspondentes aos pontos iluminados. No âmbito deste trabalho foi utilizado o princípio da triangulação considerando a intersecção de raios

do projector e da câmara. O formalismo matemático adoptado é o de [10].

3.2.1 Triangulação por Intersecção Raio-Raio

Para se aplicar o princípio da triangulação na prática é necessário conhecer a transformação rígida, dada pelo vector de translação $\mathbf{T} \in \mathbb{R}^3$ e pela matriz de rotação $\mathbf{R} \in \mathbb{R}^{3\times3}$, que determinam a posição e orientação relativas entre os sistemas de coordenadas do projector e da câmara. Isto é conseguido realizando a *calibração extrínseca* do projector e da câmara. No entanto, é necessário proceder previamente à *calibração intrínseca* do projector e da câmara para determinar os seus *parâmetros intrínsecos*. Somente desta forma, é possível determinar a equação dos raios ópticos do projector e da câmara. Considerando o modelo *pinhole* completo, um ponto $P^{(W)}$ com coordenadas WXYZ, é projectado num pixel $q^{(O)}$ da imagem com coordenadas OUV (equação 3.12). A partir do pixel $q^{(O)}$ é possível definir uma recta que contém este pixel e que passa pelo centro de projecção C, segundo a direcção dada pelo versor s ou t (figura 3.2).

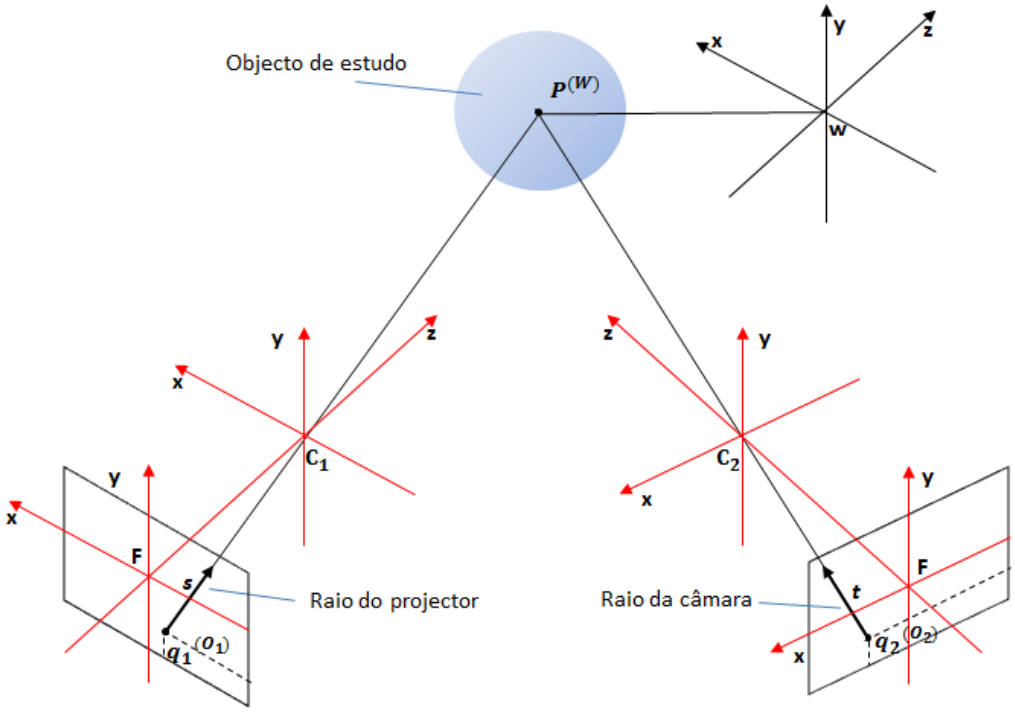


Fig. 3.2: Triangulação Raio-Raio.

Uma vez que a recta contém o centro de projecção C, a projecção de todos os pontos que ela contém possui as mesmas coordenadas no plano da imagem. Como $P^{(W)}$ é um dos pontos contidos nesta recta, é possível relacionar as coordenadas

no plano da imagem OUV do pixel q, com as suas coordenadas no mundo WXYZ. Uma vez que a matriz $\mathbf{R}$ é uma matriz de rotação, tem-se $\mathbf{R}^{-1} = \mathbf{R}^T$ e é possível determinar a equação da recta na forma paramétrica a partir da equação 3.11:

$$P^{(W)} = -\mathbf{R}^T\mathbf{T} + \lambda(\mathbf{R}^T\mathbf{A}^{-1}q^{(O)}), \tag{3.14}$$

onde $-\mathbf{R}^T\mathbf{T}$ corresponde às coordenadas WXYZ do pixel q contido no plano da imagem, e $(\mathbf{R}^T\mathbf{A}^{-1}q^{(O)})$ corresponde às coordenadas WXYZ do versor s ou t. As equações das rectas na figura 3.2 podem ser reescritas numa forma mais simples como:

$$\begin{cases} P = q_1 + \lambda_1 s \\ P = q_2 + \lambda_2 t \end{cases} \tag{3.15}$$

Considerando um *scanner* 3D composto por um projector e por uma câmara, a intersecção de um raio do projector com um raio da câmara contém um ponto (figura 3.2) P, tal que:

$$q_1 + \lambda_1 s = q_2 + \lambda_2 t \tag{3.16}$$

Uma vez que os raios podem não se intersectar, é definida uma intersecção aproximada, dada pelo ponto P mais próximo a ambas as rectas, que minimiza a soma do quadrado das distâncias a cada recta:

$$\phi(P, \lambda_1, \lambda_2) = \|q_1 + \lambda_1 s - P\|^2 + \|q_2 + \lambda_2 t - P\|^2 \tag{3.17}$$

Seja $P_1 = q_1 + \lambda_1 s$, um ponto no raio definido pela câmara e $P_2 = q_2 + \lambda_2 t$, um ponto no raio definido pelo projector, assume-se que P é o ponto médio do segmento de recta que une P_1 e P_2, como:

$$P = P_1 + \frac{1}{2}(P_2 - P_1) = P_2 + \frac{1}{2}(P_1 - P_2) \tag{3.18}$$

O passo principal consiste em determinar os parâmetros λ_1 e λ_2 que permitam determinar os pontos P_1 e P_2, mais aproximados da região de intersecção das rectas. Através do formalismo matemático deduzido em [10], considerando os parâmetros *íntrísecos* e *extrínsecos* do projector e da câmara, são então determinados os parâmetros λ_1 e λ_2 que permitam determinar a posição dos pontos P_1 e P_2 ao longo das rectas. A expressão deduzida é dada por:

$$\begin{pmatrix}\lambda_1\\ \lambda_2\end{pmatrix} = \frac{1}{\|s\|^2\|t\|^2 - (s^t t)^2}\begin{pmatrix}\|t\|^2 & (s^t t)^2\\ t^t s & \|s\|^2\end{pmatrix}\begin{pmatrix}s^t(q_2-q_1)\\ t^t(q_1-q_2)\end{pmatrix} \tag{3.19}$$

No âmbito deste trabalho, a triangulação óptica pela intersecção raio-raio foi aplicada recorrendo a um pacote de código-fonte *open-source* desenvolvido por D. Lanman e G. Taubin [10] e escrito em C++. O seguinte excerto de código apresenta o método utilizado, denominado *intersectLineWithLine3D*.

```
void intersectLineWithLine3D(const float* q1,
 const float* s,
 const float* q2,
 const float* t,
 float* p){

// Cálculos intermédios.
float q12[3],
s_dot_s = 0,
t_dot_t = 0,
s_dot_t = 0,
q12_dot_s = 0,
q12_dot_t = 0;

for(int i=0; i<3; i++){
  q12[i]      =  q1[i]-q2[i];
  s_dot_s  +=  s[i]*s[i];
  t_dot_t  +=  t[i]*t[i];
  s_dot_t  +=  s[i]*t[i];
  q12_dot_s +=q12[i]*s[i];
  q12_dot_t +=q12[i]*t[i];
}

// Cálculo de lambda1 e lambda2.
float lambda1, lambda2, denom;
denom = s_dot_s*t_dot_t - s_dot_t*s_dot_t;
lambda1 =  (s_dot_t/denom)*q12_dot_t - (t_dot_t/denom)*q12_dot_s;
lambda2 = -(s_dot_t/denom)*q12_dot_s + (s_dot_s/denom)*q12_dot_t;

// Determinação do ponto P mais próximo.
for(int i=0; i<3; i++)
  p[i] = ( (q1[i]+lambda1*s[i]) + (q2[i]+lambda2*t[i]) )/2;
}
```

Capítulo 4

Implementação de um *Scanner* 3D

Foi implementado um *scanner* 3D baseado na projecção de luz estruturada, codificada temporalmente. Baseado na estrutura de um *scanner* 3D activo, este sistema foi constituído por um emissor de luz (projector de vídeo), um receptor de luz (câmara de vídeo), e um sistema de controlo (computador). Pretendeu-se desenvolver o sistema o mais automático possível, de forma a minimizar a interacção utilizador-sistema. Para tal, foi introduzida uma plataforma para permitir a rotação automática do objecto em estudo e o controlo da sua posição angular no *scanner* 3D. A plataforma constitui ainda outra vantagem significativa, uma vez que torna desnecessário o reposicionamento de objectos frágeis e sensíveis, cuja integridade possa ser comprometida. A câmara, o projector e o computador utilizados no *scanner* 3D são componentes *"off-the-shelf"*, enquanto que a plataforma rotativa foi desenvolvida no âmbito deste trabalho. A implementação do *scanner* 3D também compreendeu uma componente de desenvolvimento de software, na criação de um conjunto de algoritmos para o controlo e para a calibração do sistema.

4.1 *Hardware* Utilizado

4.1.1 Câmara

A câmara de vídeo utilizada foi uma Logitech Webcam Pro 9000 (figura 4.1, (a)). A escolha recaiu sobre este modelo uma vez que apresenta uma boa relação qualidade/preço, possui boas características em termos de hardware, e é compatível com o software e com os *toolkits* utilizados. A tabela 4.1 apresenta as especificações da câmara de vídeo.

4.1.2 Projector de vídeo

O projector de vídeo utilizado foi um LG RD-JT31 (figura 4.1, (b)). No caso dos projectores, qualquer um pode ser utilizado, desde que o sistema operativo o reconheça como um monitor adicional, e seja adequado às condições de luz ambiente. A tabela 4.2 apresenta as especificações do projector.

4.1.3 Plataforma Rotativa

A plataforma rotativa foi desenvolvida integralmente no âmbito deste trabalho para automatizar o funcionamento do *scanner* 3D. Segue-se uma referência à sua estrutura mecânica e ao circuito electrónico de controlo. O material utilizado no desenvolvimento da plataforma rotativa encontra-se enunciado no Apêndice A, na secção A.1.

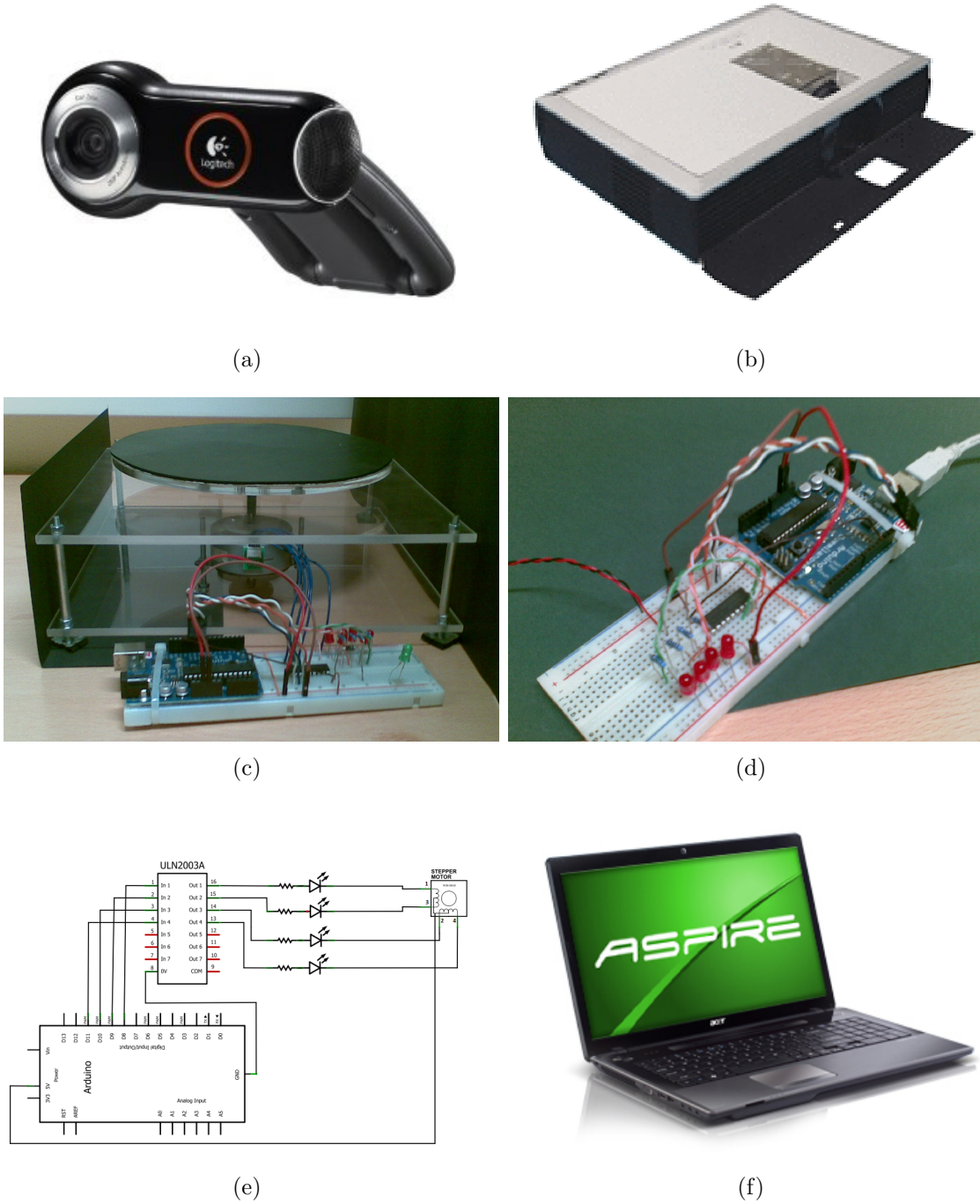
Estrutura Mecânica

A estrutura da plataforma rotativa (figura 4.1, (c)), é constituída por dois componentes: uma base, responsável por gerar o binário para a rotação, e uma superfície para colocar o objecto a analisar. A base da plataforma contém um motor de passos (48 passos e 7,5º/passo) integrado com uma roda dentada e um eixo de rotação. Devido à engrenagem na plataforma, cada passo do motor provoca uma rotação de 1.24º (360º/290 passos) na superfície. O motor encontra-se ligado ao circuito electrónico de controlo junto à base da plataforma rotativa. A superfície assenta no topo do eixo de rotação. A estrutura do *scanner* foi desenvolvida em acrílico, usando parafusos para unir os diversos componentes. Para minimizar as fontes de erro durante a aquisição dos pontos, o topo da superfície e a face da plataforma rotativa voltada para a câmara e para o projector foram revestidos a preto. As dimensões da plataforma rotativa são as seguintes:

- Base: (250 x 250 x 100) mm;
- Prato: 200 mm de diâmetro.

Electrónica de Controlo

Foi desenvolvido um circuito electrónico para permitir o controlo do motor de passos a partir do computador, de forma a originar a rotação pretendida na plataforma rotativa (figura 4.1, (d) e (e)). Para controlar o motor foi utilizada uma placa



(a) (b)

(c) (d)

(e) (f)

Fig. 4.1: Elementos constituíntes do *scanner* 3D. (a) Logitech Webcam Pro 9000, (b) Projector LG RD-JT31, (c) Estrutura mecânica da plataforma rotativa, (d) e (e) Electrónica de controlo, (f) Computador Acer Aspire 5740G.

Arduino duemilanove, juntamente com um driver ULN2003. O Arduino recebe instruções do computador por comunicação série, através da porta USB e envia

sinais pelos seus pinos digitais 8, 9, 10 e 11 para as entradas 1, 2, 3 e 4 do driver ULN2003. Este, após receber a lógica de controlo para ligar ou desligar as bobinas do motor, transmite corrente para o motor pelas saídas 13, 14, 15 e 16 para os cabos 4, 2, 3 e 1 do motor. A terra do ULN2003 é ligada à terra do arduino, e o motor é alimentado pela saída de 5V do Arduino. Encontram-se 4 LEDs e 4 resistências de 1kΩ no circuito, entre o ULN2003 e o motor, para indicarem quais as bobinas que estão ligadas.

4.1.4 Computador

Utilizou-se um computador portátil Acer Aspire 5740G (figura 4.1, (f)). Poder-se-ia ter utilizado qualquer computador que respeitasse os requisitos exigidos pelos componentes de *software* e *toolkits* usados (enunciados no Apêndice A, na secção A.2). As características do computador utilizado encontram-se na tabela 4.3.

Tab. 4.1: Câmara Logitech Pro 9000.

Especificações	
Resolução	2 megapixel
Tipo de sensor óptico	CMOS
Profundidade de cor	24-bit, True color
Captura de vídeo	1600 x 1200 pixels, 30 frames/s
Interface de ligação	USB 2.0

Tab. 4.2: Projector LG RD-JT31.

Especificações	
Resolução	1024x768 pixeis
Contraste	800:1
Luminosidade	1100 ANSI Lumen
Interface de ligação	S-video/USB/VGA/Composite video

Tab. 4.3: Computador Acer Aspire 5740G.

Especificações	
Processador	Intel Core i7 @ 2.66 GHz
Memória RAM	4 GB DDR3
Disco Rígido	640 GB, 7200 rpm
Placa gráfica	Radeon HD 5650, 1 GB DDR3

4.2 Montagem Experimental

O esquema de ligação e de posicionamento dos componentes do *scanner* 3D encontra-se na figura 4.2. A câmara liga-se a uma entrada USB do computador, e o projector liga-se à entrada VGA de 15 pinos do computador. O projector é configurado nas definições de visualização do *Windows* como ecrã secundário, alinhado ao topo e à esquerda do ecrã principal. O circuito electrónico da plataforma rotativa é ligado a uma porta USB do computador através da interface USB do Arduino. Como forma de minimizar os erros no *scanner* 3D por reflexão, é incluido um fundo preto: o *background*.

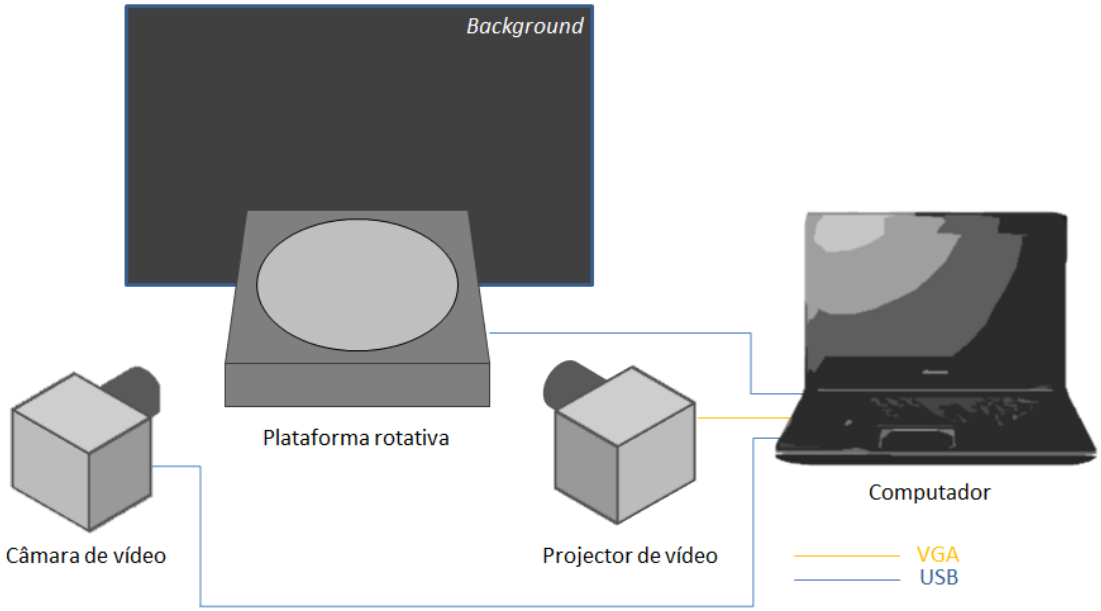


Fig. 4.2: Esquema de montagem do *scanner* 3D.

Na montagem efectuada, a distância entre a câmara de vídeo e as extremidades da plataforma rotativa foi de 750 mm e 1000 mm, respectivamente. Esta distância é igual para o projector, como se pode verificar na figura 4.3.

4.3 *Software* Desenvolvido

A componente de desenvolvimento de *software* compreendeu, em primeiro lugar, a programação do Arduino para receber instruções do computador e transmitir sinais para controlo do motor. Isso foi conseguido através do desenvolvimento de um algoritmo chamado *Stepper*, carregado para o microcontrolador ATmega328 no Arduino.

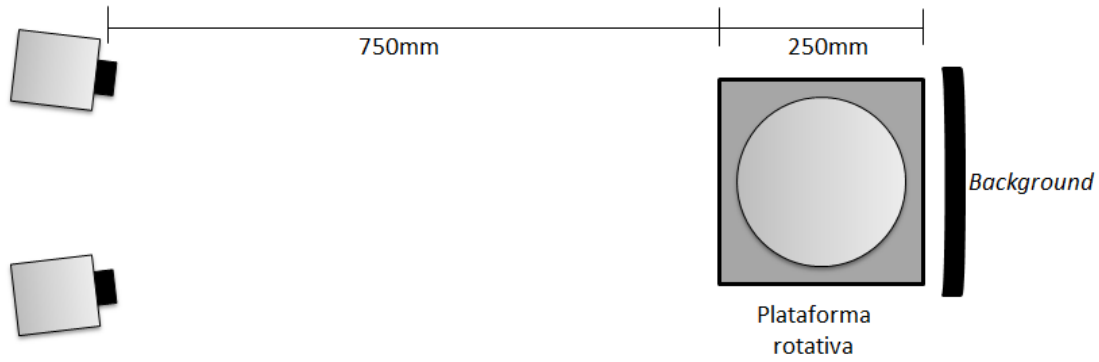


Fig. 4.3: Montagem experimental do *scanner*3D.

Seguidamente, foi implementado um algoritmo para testar o motor de passos e um algoritmo para adquirir pontos 3D tirando partido da plataforma rotativa. Todo o conjunto de *software* e *toolkits* utilizados ao longo do trabalho encontra-se enunciado no Apêndice A, na secção A.2.

4.3.1 Stepper

O algoritmo *Stepper* permite ao Arduino receber um comando do computador através de comunicação por porta série, usando a sua interface USB, e enviar sinais para controlar o motor de passos. Este algoritmo foi desenvolvido utilizando a IDE Arduino alpha 0021, recorrendo à biblioteca *"Stepper"* que permite o controlo de motores de passos unipolares ou bipolares. O comando é enviado para o Arduino numa *string* constituída por cinco caracteres, cada um com uma função definida na tabela 4.4:

Tab. 4.4: Caracteres enviados para o Arduino, para controlo do motor de passos.

Caracter	Valor	Função
0	'X'	Selecionar o motor
1	'-' ou outro valor	Se for '-', a rotação é feita no sentido negativo.
2 3 e 4	Qualquer inteiro	Define o n.º de passos que o motor vai dar.

Por exemplo, o comando "X-123"corresponde a uma rotação de 123 passos no sentido negativo, enquanto que o comando "X+001"corresponde a uma rotação de 1 passo no sentido positivo. O código desenvolvido foi descarregado para o Arduino através da IDE Arduino alpha e foi armazenado na memória do microcontrolador ATmega 328. O Arduino envia os sinais de controlo através dos seus pinos digitais 8, 9, 10 e 11.

4.3.2 cvStructuredLight

cvStructuredLight é um pacote de código-fonte desenvolvido por Douglas Lanman e Gabriel Taubin [10] para a implementação de um *scanner* 3D de luz estruturada. O código foi escrito em MATLAB e C++, usando OpenCV, e é livre para ser usado e alterado. A versão do código utilizada neste trabalho foi escrita em C++ e encontra-se disponível para download em: `http://mesh.brown.edu/byo3d/source.html`. O código-fonte suporta *scanners* 3D constituídos por um projector e por uma câmara, e define um conjunto de funções para:

- Aquisição tridimensional de pontos;
- Estimação do *background*;
- *Reset* do *background*;
- Calibração intrínseca da câmara;
- Calibração intrínseca do projector;
- Calibração extrínseca da câmara e do projector;
- Calibração da câmara e do projector simultaneamente.

Na figura 4.4 é possível verificar o ecrã principal definido no código-fonte, onde se encontram as opções disponíveis.

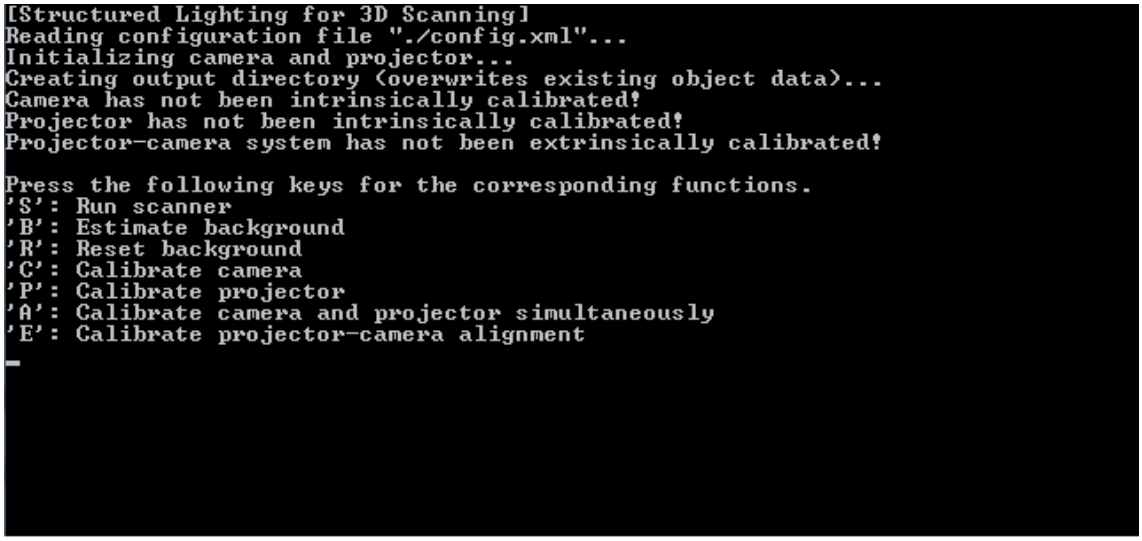


Fig. 4.4: Ecrã principal *cvStructuredLight*.

A aquisição de pontos efectua-se através da projecção e da captura de uma sequência de padrões binários de luz estruturada utilizando código Gray (secção 2.3.2), e fornece à saída nuvens de pontos do tipo VRML (*Virtual Reality Modeling Language*). A sequência de imagens capturadas pela câmara é descodificada para resolver as correspondências entre cada pixel na câmara e cada linha ou coluna do projector. Sendo conhecidos os parâmetros intrínsecos e extrínsecos do projector e da câmara, bem como as correspondências, os pontos 3D são adquiridos por triangulação óptica.

A calibração do sistema utiliza funções básicas de OpenCV para a detecção de padrões de xadrez e para calibração baseada em homografia, segundo o método de Zhang [42]. A calibração da câmara é realizada obtendo uma sequência de duas ou mais imagens de um padrão de xadrez de calibração, sob diferentes orientações. As dimensões do padrão de xadrez são conhecidas. Desta forma, cada imagem obtida permite estabelecer um conjunto de correspondências 2D-3D, fazendo um mapeamento entre as coordenadas no plano da imagem e os pontos no padrão de xadrez. Assim, os parâmetros intrínsecos da câmara são resolvidos separadamente. A calibração do projector é realizada utilizando uma sequência de duas imagens. Em primeiro lugar, o padrão de xadrez é detectado automaticamente usando a câmara. De seguida, o projector emite um padrão de xadrez virtual e a primeira imagem é subtraída da segunda. São então utilizadas as funções de OpenCV para detectar os cantos do padrão de xadrez virtual. Uma calibração prévia da câmara permite que as coordenadas 3D de cada canto do padrão virtual sejam reconstruídas. Por fim, o conjunto de correspondências 2D-3D, mapeando os pixels 2D do projector em pontos 3D no plano de calibração, é utilizado para avaliar a calibração intrínseca do projector. Posteriormente, um único par de imagens, compreendendo o padrão de xadrez físico e o padrão de xadrez virtual projectado sobre ele, é utilizado para estimar a calibração extrínseca do projector e da câmara. O procedimento para a calibração do sistema (projector e câmara) será descrito em pormenor no capítulo 5.

Parametrização do sistema

Os parâmetros de entrada para as diferentes funções são definidos num ficheiro de configuração do tipo XML, chamado "config.xml". A tabela 4.5 apresenta os parâmetros definidos no ficheiro "config.xml". Alguns dos parâmetros são auto-explicativos, enquanto que outros serão descritos mais detalhadamente. Os valores

definidos para cada parâmetro correspondem aos valores utilizados neste trabalho. Em primeiro lugar foi definido o directório de saída da aplicação como "./output". O nome do objecto, definido como "teste", define o subdirectório onde serão armazenadas as saídas da aplicação relativas a esse mesmo objecto. A opção para salvar resultados intermédios permite activar ou desactivar o armazenamento de resultados intermédios como imagens, mapas de profundidade e ficheiros temporários.

Os parâmetros do projector e da câmara foram introduzidos considerando os modelos utilizados no trabalho (descritos na secção 4.1). Seguem-se as características dos padrões de xadrez utilizados na calibração do projector e da câmara. Um dos padrões de xadrez é físico, e foi impresso em cartolina, enquanto que o outro padrão de xadrez é virtual e projectado pelo projector de vídeo.

Relativamente à aquisição de pontos, a aplicação disponibiliza duas formas de triângulação óptica - triangulação raio-plano e triangulação raio-raio (secção 3.2.1)-, e três modos diferentes para a codificação dos padrões de luz projectados: codificação das linhas do projector, codificação das colunas do projector, e codificação tanto das linhas como das colunas do projector (secção 2.3.1). No âmbito deste trabalho foi seleccionada a triangulação raio-raio e a codificação de linhas e colunas do projector. O campo *field delay* controla a sincronização entre a câmara e o projector. Uma vez que não existe nenhum botão para captura dos padrões de luz estruturada, a câmara e o projector funcionam em *open loop*. Assim, é necessário definir o intervalo de tempo que o sistema vai dar entre a projecção de um padrão de luz estruturada e a captura da imagem pela câmara. Esse intervalo foi definido no campo *field delay* como 300 ms, correspondendo ao valor por defeito. O *Threshold* mínimo de contraste permite definir a variação mínima entre um plano claro e um plano escuro de forma ao sistema determinar se um ponto está ou não iluminado pelo projector. O valor introduzido corresponde ao valor por defeito. As distâncias mínima e máxima definem os planos de corte proximal e distal, respectivamente (ver figura 4.3). Estes planos limitam a região a ser digitalizada pelo *scanner*, fazendo com que os pontos adquiridos fora dessa região, não sejam incluídos nas nuvens de pontos. O valor da máxima variação de distância é o valor por defeito. Por fim, foi definida a resolução da janela de visualização da aplicação em 640×480 pixels.

4.3.3 myStructuredLight

Para se testar o código-fonte procedeu-se à sua compilação. O sistema utilizado para testes foi constituído pelos elementos descritos na secção 4.1 e foi adoptado o

Tab. 4.5: Parametrização do sistema

Parâmetro	Valor
Directório para armazenar a saída do *scanner*	"./output"
Nome do objecto	"teste"
Salvar resultados intermédios	Não
Resolução da câmara	800 x 600 pixéis
Resolução do projector	1024 x 768 pixéis
Ganho do projector e da câmara	50
Modelo de distorção do projector e da câmara	Tangencial
Padrão de xadrez físico	
N.º de cantos interiores horizontais	8
N.º de cantos interiores verticais	6
Dimensão dos quadrados	29 x 29 mm
Padrão de xadrez projectado	
N.º de cantos interiores horizontais	8
N.º de cantos interiores verticais	6
Dimensão dos quadrados	75 x 75 pixéis
Aquisição de pontos	
Tipo de projecção	Planos verticais e horizontais
Triangulação	Raio-Raio
Frame delay	300 ms
Threshold mínimo de contraste	50
Distância mínima entre o *scanner* o objecto	750 mm
Distância máxima entre o *scanner* o objecto	1000 mm
Máxima variação de distância	100 mm
Distância mínima ao *background*	20 mm
Resolução da janela de visualização da aplicação	640 x 480 pixéis

esquema de montagem apresentado na secção 4.2. No entanto, o código não contemplava a plataforma rotativa e, após a realização dos testes verificou-se que, para se poder implementar um *scanner* 3D automático e tirar partido da plataforma rotativa, teriam que ser feitas alterações ao código. Por esse motivo, foram desenvolvidas e acrescentadas duas novas funcionalidades ao código original:

- ***Send command to Stepper***, que permite testar o motor de passos na plataforma rotativa;

- ***Scan rotating object***, que permite a aquisição de pontos tirando partido da plataforma rotativa.

Send command to Stepper

Send command to Stepper permite testar o motor de passos através do envio de um comando por comunicação série para o Arduino. O comando é enviado numa *string* introduzida pelo utilizador, e deve respeitar os parâmetros na tabela 4.4 de modo a provocar a rotação pretendida. Esta funcionalidade foi desenvolvida para testes.

Scan rotating object

Scan rotating object permite adquirir pontos de diferentes regiões de um objecto, através da introdução de uma rotação entre aquisições de pontos sucessivas. Esta rotação é conseguida através do envio do comando "X+005"para o Arduino por comunicação série, no final de cada aquisição de pontos (ver tabela 4.4). Este comando faz com que o motor dê 5 passos, provocando uma rotação θ de 6.20º (5×1.24º), na plataforma rotativa devido à engrenagem montada (figura 4.5). Graças à rotação θ, introduzida entre cada aquisição de pontos, o objecto apresenta uma rotação total ϕ, face à sua posição incial, dada pela equação 4.1,

$$\phi_n = (n-1) \times \theta, \tag{4.1}$$

na qual n corresponde a um ID (número inteiro), que identifica cada aquisição de pontos. Para se adquirirem pontos de toda a superfície visível de um objecto são necessárias 58 aquisições de pontos ($58 \times 6.20° \approx 360°$).

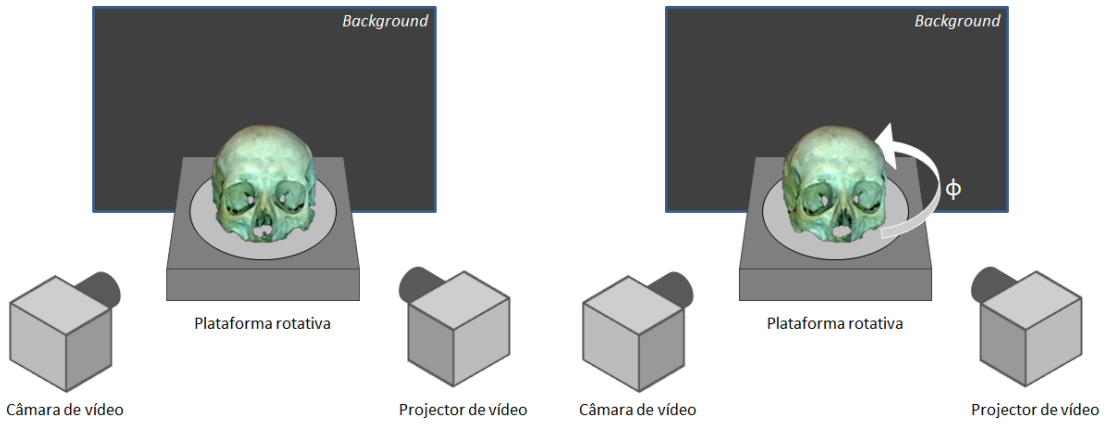


Fig. 4.5: Rotação no objecto de estudo ao longo de aquisições de pontos, para se adquirirem pontos de diferentes regiões do objecto. (a) Aquisição 1, ($\phi_1 = 0$) e (b) Aquisição 2, ($\phi_2 = \theta$).

Cada aquisição de pontos origina um *view*, ou seja, uma nuvem de pontos que corresponde a um ponto de vista do objecto. Cada *view* é identificado por um ID igual ao da aquisição de pontos correspondente. Assim, a aquisição de pontos 1 origina o *view* 1, e a aquisição de pontos 2, origina o *view* 2. Ao conjunto de várias aquisições de pontos corresponde um *scan*, também identificado por um ID. Para um dado objecto podem ser realizados vários *scans*, sendo aquiridos em cada um, vários *views* desse objecto (figura 4.6).

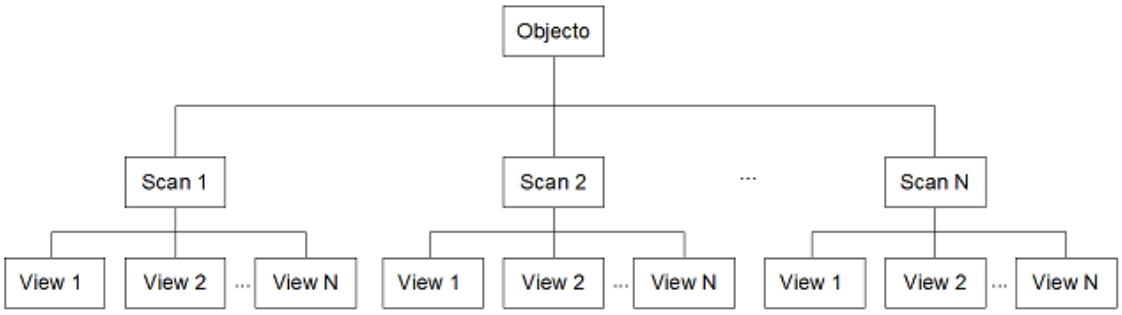


Fig. 4.6: Organização das aquisições de pontos.

Scan rotating object recebe como parâmetros de entrada, o número pretendido de aquisições de pontos (*View max*), e os parâmetros no ficheiro "config.xml"(ver tabela 4.5). As nuvens de pontos adquiridas são armazenadas em ficheiros do tipo VRML, com a extensão "*.wrl*". O nome atribuído a cada ficheiro de saída resulta da concatenação do nome do objecto (definido no ficheiro "config.xml") com o ID do *scan* e o ID do *view* correspondentes. Assim, por exemplo, o ficheiro de saída correspondente ao objecto "teste", ao *scan* 1 e ao *view* 3, é guardado num ficheiro com o nome "teste_01_003.wrl". A figura 4.7 apresenta a *pipeline* da opção *Scan rotating object*. *Scan rotating object* traduz-se numa enorme vantagem já que, para se adquirir pontos de diferentes regiões de um objecto usando a funcionalidade original - *Run Scanner*-, seria necessário seleccionar esta opção e reposicionar o objecto várias vezes. Este processo requiriria uma elevada interacção por parte do utilizador e tornaria o sistema mais vulnerável a erros. O código foi compilado e originou-se a aplicação *myStructuredLight*, sendo possível ver o seu menu principal na figura 4.8.

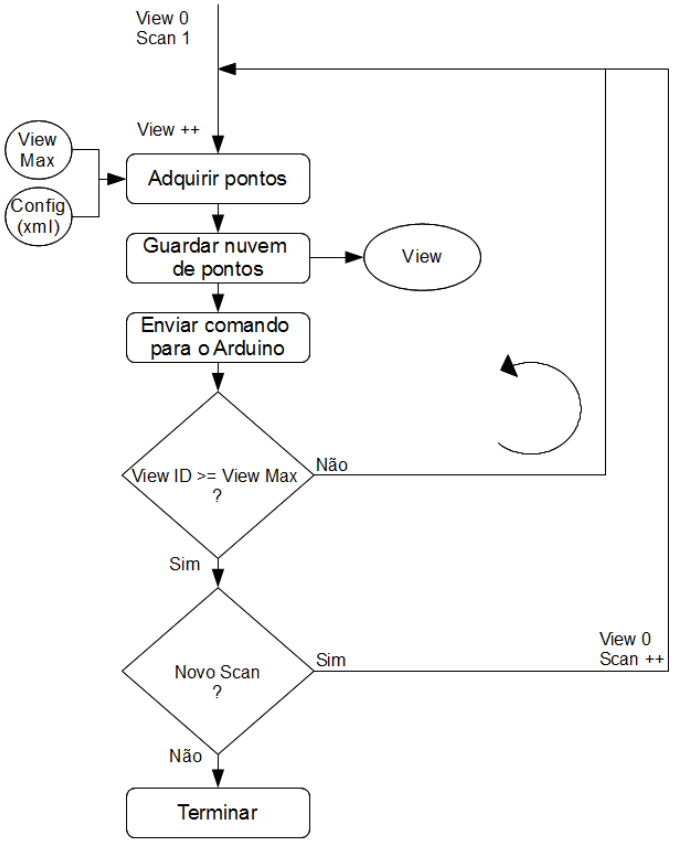


Fig. 4.7: *Pipeline Scan rotating object.*

```
[Structured Lighting for 3D Scanning]
Reading configuration file "./config.xml"...
Initializing camera and projector...
Creating output directory (overwrites existing object data)...
Camera has not been intrinsically calibrated!
Projector has not been intrinsically calibrated!
Projector-camera system has not been extrinsically calibrated!

Press the following keys for the corresponding functions.
'S': Run scanner
'B': Estimate background
'R': Reset background
'C': Calibrate camera
'P': Calibrate projector
'E': Calibrate projector-camera alignment
------------------------
'1': Send command to stepper
'2': Scan rotating object
------------------------
'ESC': Exit application
```

Fig. 4.8: Ecrã principal *myStructuredLight*.

Como é possível verificar, todas as restantes funcionalidades definidas no código original foram preservadas.

Capítulo 5

Aquisição de Nuvens de Pontos

Neste capítulo será abordada a aquisição de pontos utilizando o *scanner* 3D de luz estruturada implementado no capítulo 4. Antes de se adquirirem pontos, é necessário proceder à calibração do *scanner*. Somente desta forma é possível reconstruir os pontos por triangulação óptica entre o projector e a câmara.

5.1 Calibração do Sistema

Nesta secção serão abordados aspectos relevantes sobre a calibração do projector e da câmara, e será descrito o procedimento utilizado na calibração do *scanner* desenvolvido.

5.1.1 Calibração Intrínseca da Câmara

A calibração da câmara é o processo que permite determinar os parâmetros que caracterizam a sua geometria interna e as suas características ópticas (*parâmetros intrínsecos*, ou matriz **A** (secção 3.1.2, eq. 3.8 e 3.9), bem como o posicionamento da câmara relativamente ao sistema de coordenadas do mundo (*parâmetros extrínsecos*). A calibração da câmara permite:

- Determinar as coordenadas 2D no plano da imagem, de um ponto no espaço 3D, conhecendo as suas coordenadas (x, y, z);
- Dadas as coordenadas 2D de um ponto da imagem, determinar a equação da recta 3D que passa por esse ponto no plano da imagem e pelo centro óptico da câmara (figura 5.1).

Fig. 5.1: Modelo da câmara.

Para calibrar a câmara tira-se partido da funcionalidade *Calibrate Camera* na aplicação *myStructuredLight*. A calibração da câmara baseia-se no método proposto por Zhang [42], que é largamente utilizado no meio da visão por computador devido à sua simplicidade de execução. Este método requer que a câmara observe um padrão planar de calibração, constituído por um conjunto de características 3D conhecidas, sob diferentes orientações (pelo menos duas), sendo que tanto a câmara como o padrão de calibração podem ser livremente posicionados. As imagens obtidas pela câmara permitem assim o estabelecimento de um conjunto de correspondências 2D-3D, mapeando as coordenadas dos pontos no plano de imagem da câmara, com a posição dos pontos no espaço 3D [42]. Desta forma são obtidos os parâmetros intrínsecos da câmara. O procedimento para a calibração da câmara é o seguinte:

- Imprime-se um padrão de xadrez com número de quadrados conhecido, e com dimensões conhecidas, e afixa-se numa superfície planar rija;
- Defininem-se as características do padrão de calibração e da câmara no ficheiro "config.xml"de acordo com a tabela 4.5;
- No *driver* da câmara de vídeo desactiva-se a auto-exposição, o balanço de brancos e a focagem automática;
- Utilizando a aplicação *myStructuredLight* selecciona-se a opção *Calibrate camera* pressionando a tecla 'C';
- Introduz-se o número de imagens a capturar;
- Sempre que a câmara detectar os cantos dos quadrados é apresentado um conjunto de linhas coloridas no ecrã. Nessa altura pressiona-se a tecla 'N' para capturar uma imagem;

- Entre cada captura altera-se a orientação do padrão de xadrez.

Para a calibração da câmara imprimiu-se um padrão de xadrez em cartolina, que foi afixado numa superfície planar rija. O padrão era constituído por 9×7 quadrados, cada um com 29×29 mm. O número de imagens a capturar no processo de calibração correspondeu a 15. A figura 5.2, apresenta um conjunto de imagens do padrão de xadrez capturadas no processo de calibração da câmara.

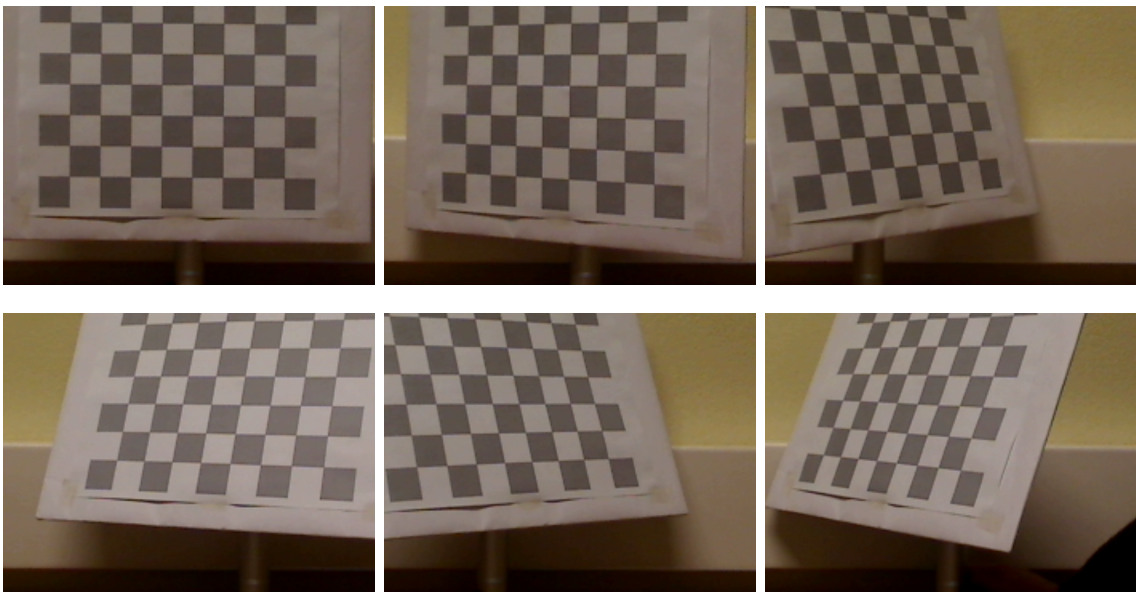


Fig. 5.2: Calibração da câmara. Conjunto de imagens capturadas, correspondendo a 6 de 15 diferentes orientações do padrão de xadrez usado para a calibração da câmara.

No final da calibração, obteve-se a seguinte matriz $\mathbf{A}$, de *parâmetros intrínsecos* da câmara:

$$\mathbf{A} = \begin{pmatrix} 2218 & 0 & 490 \\ 0 & 2229 & 243 \\ 0 & 0 & 1 \end{pmatrix} \tag{5.1}$$

No final da sequência de captura, os resultados da calibração são armazenados no directório "calib\cam"dentro da pasta "\output", como definido no ficheiro "config.xml".

5.1.2 Calibração Intrínseca do Projector

A calibração do projector é realizada após a calibração da câmara, usando sequências de duas imagens. Em primeiro lugar, o padrão de xadrez físico é detectado pelo sis-

tema, tal como na calibração da câmara. Seguidamente, um padrão de xadrez virtual é projectado. A primeira imagem é subtraída da segunda. As funções do OpenCV para a detecção de padrões de xadrez permitem encontrar os cantos do padrão virtual. A calibração prévia da câmara permite a reconstrução das coordenadas 3D de cada canto do padrão de xadrez virtual. O conjunto completo de correspondências 2D-3D, mapeando os pixels 2D do projector em pontos 3D no padrão de calibração, é utilizado para avaliar a calibração intrínseca do projector. O procedimento para a calibração do projector é o seguinte:

- Definem-se as características do projector e do padrão projectado, no ficheiro "config.xml"de acordo com a tabela 4.5;
- Utilizando a aplicação *myStructuredLight* selecciona-se a opção *Calibrate projector* pressionando a tecla 'P';
- Introduz-se o número de imagens que se pretende capturar;
- Sempre que é detectado o padrão de xadrez físico, o projector emite o padrão de xadrez virtual, e no ecrã são apresentadas linhas coloridas. Nesta altura pressiona-se a tecla 'N' para capturar uma imagem;
- Entre cada captura de imagem é alterada a orientação do padrão de xadrez.

Neste trabalho, o número de imagens capturadas correspondeu a 15. A figura 5.3 apresenta um conjunto de imagens capturadas no processo de calibração do projector.

No final da calibração, obteve-se a seguinte matriz $\mathbf{A}$, de *parâmetros intrínsecos* do projector:

$$\mathbf{A} = \begin{pmatrix} 2255 & 0 & 6373 \\ 0 & 2239 & 9150 \\ 0 & 0 & 1 \end{pmatrix} \tag{5.2}$$

Os resultados da calibração do projector são gravados no directório "calib\proj"na pasta "\output", como definido no ficheiro "config.xml".

5.1.3 Calibração Extrínseca do Projector e da Câmara

Por fim, procede-se à calibração extrínseca do projector e da câmara. Desta forma determina-se a transformação que relaciona a posição e a orientação do sistema de coordenadas da câmara com a posição e a orientação do sistema de coordenadas do

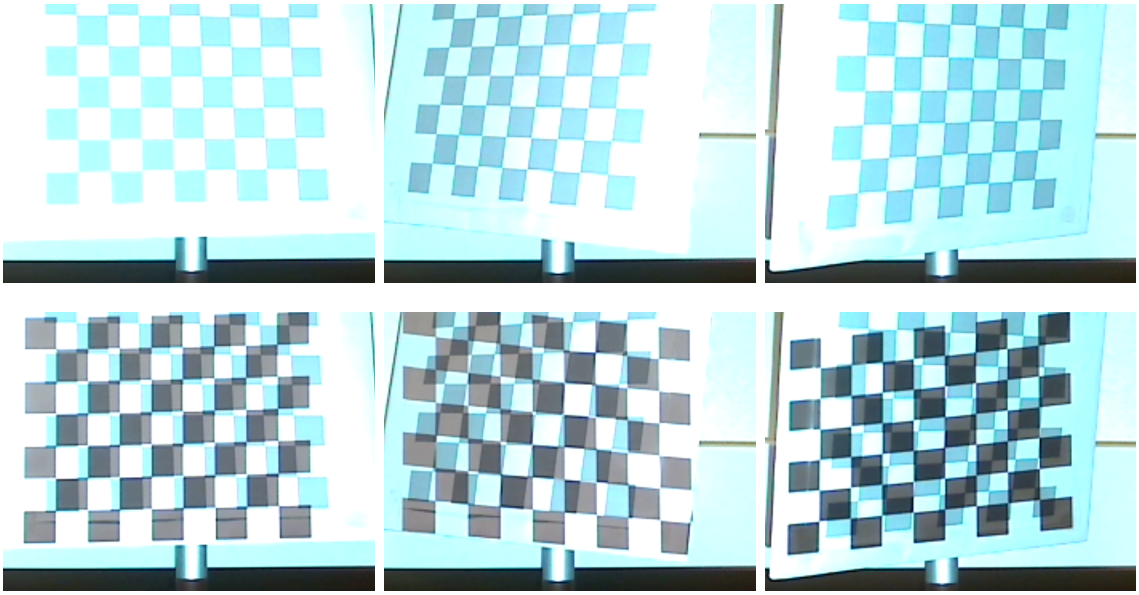


Fig. 5.3: Calibração do projector. Conjunto de pares de imagens utilizados na calibração do projector. A primeira linha apresenta o padrão de xadrez visto pela câmara. A segunda linha apresenta o padrão de xadrez virtual projectado.

projector (secção 3.1.2). Um simples par de imagens, compreendendo o padrão de xadrez físico e o padrão de xadrez virtual projectado na sua superfície, é utilizado para estimar a calibração extrínseca do projector e da câmara. O procedimento para a calibração extrínseca do projector e da câmara é o seguinte:

- Utilizando a aplicação *myStructuredLight* selecciona-se a opção *Calibrate projector-camera alignment* premindo a tecla 'E';
- O projector emite o padrão de xadrez virtual e a câmara de vídeo procede à sua captura.

Terminada a calibração extrínseca, os resultados são gravados no directório "calib\proj"dentro da pasta "\output", como definido no ficheiro "config.xml".

5.2 Aquisição de uma Nuvem de Pontos

Com o *scanner* 3D montado e calibrado é possível proceder à aquisição de pontos. Note-se que se o posicionamento do projector e/ou câmara for alterado após a calibração extrínseca, é necessário calcular a nova transformação entre estes antes de

adquirirem pontos. Se o objectivo for a aquisição de somente uma nuvem de pontos, o procedimento a adoptar é o seguinte:

1. Inicia-se a aplicação *myStructuredLight* e selecciona-se a funcionalidade *Estimate background*, premindo a tecla 'B' no menu principal. Esta funcionalidade estima o fundo da cena, fazendo com que, no final, os pontos adquiridos correspondentes a esse mesmo fundo sejam removidos das nuvens de pontos;

2. Selecciona-se a opção *Run scanner* no menu principal premindo a tecla 'S';

3. Posiciona-se o objecto a analisar no centro da plataforma rotativa, voltado de frente para a câmara e prime-se uma tecla aleatória, ordenando o início do processo de aquisição de pontos.

5.3 Aquisição de Várias Nuvens de Pontos

Para se adquirir várias nuvens de pontos, tirando partido da plataforma rotativa, adpota-se o seguinte procedimento:

1. Inicia-se a aplicação *myStructuredLight* e selecciona-se a funcionalidade *Estimate background*, premindo a tecla 'B' no ecrã principal;

2. Selecciona-se a opção *Scan rotating object* no menu principal da aplicação premindo-se a tecla '2';

3. Introduz-se o número de aquisições de pontos a realizar;

4. Posiciona-se o objecto no centro da plataforma rotativa, voltado de frente para a câmara e prime-se uma tecla aleatória, ordenando o início do processo de aquisição de pontos.

No âmbito deste trabalho foram realizados 2 *scans* a um crânio, sob dois posicionamentos diferentes. No primeiro *scan* o crânio foi colocado na posição vertical e no segundo *scan* foi colocado horizontalmente. Em cada *scan* foram adquiridas 58 nuvens de pontos (*views*), sendo projectada uma sequência de padrões de luz estruturada sobre o crânio em cada aquisição. Na figura 5.4 é possível observar alguns dos padrões de luz estruturada projectados no crânio. No final de cada aquisição de pontos foi originada uma nuvem com as coordenadas 3D dos pontos observados pela câmara, relativamente ao seu sistema de coordenadas.

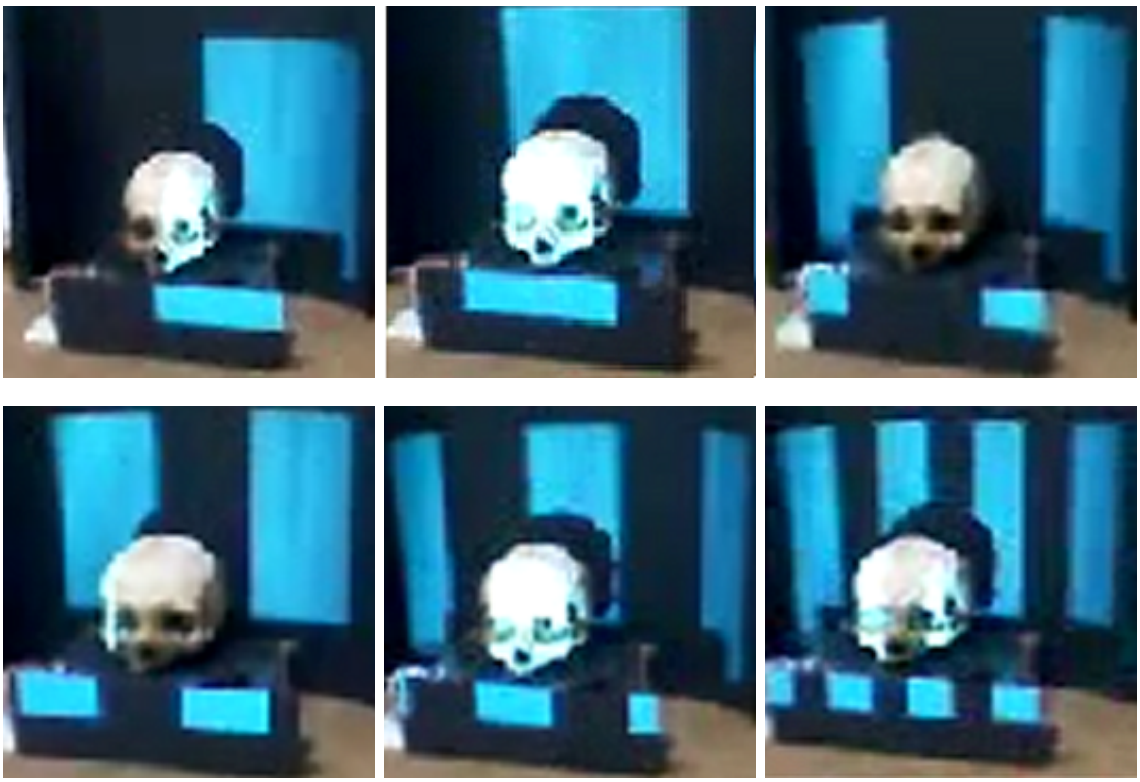


Fig. 5.4: Alguns dos padrões de luz estruturada projectados sobre o crânio.

5.4 Conversão de Formato

As nuvens de pontos adquiridas foram armazenadas no formato VRML. Este formato, definido como padrão ISO em 1997, permite descrever objectos 3D e ambientes 3D interactivos, sendo utilizado em várias áreas como a visualização científica, apresentações multimédia, páginas web, entre outros [43]. Nos últimos anos, o formato VRML tornou-se num formato padrão para transmitir ambientes 3D através da Internet, uma vez que é possível aceder à informação representada através de *browsers* web. Em [44] é possível verificar uma descrição mais detalhada do formato VRML e das suas aplicações. Cada ficheiro VRML:

- Estabelece um sistema de coordenadas do mundo para todos os objectos definidos do ficheiro, bem como para os objectos incluídos pelo ficheiro;
- Define e compõe um conjunto de objectos 3D e multimédia;
- Pode especificar hiperligações para outros ficheiros e aplicações;
- Consegue definir o comportamento dos objectos.

No âmbito deste trabalho, as nuvens de pontos foram convertidas do formato VRML para o formato VTK [45]. Esta conversão foi realizada para tornar mais simples a utilização e o processamento das nuvens de pontos, já que os algoritmos utilizados ao longo do trabalho foram implementados utilizando VTK e ITK. Foi também levada em conta a maior familiarização com o formato VTK. Assim, foi desenvolvido um algoritmo para converter os ficheiros no formato VRML em ficheiros *legacy* VTK do tipo vtkPolyData. O algoritmo foi desenvolvido utilizando VTK, e a conversão foi realizada para cada nuven de pontos - *view*-, obtida pelo *scanner* 3D. Os ficheiros VTK são constituídos por cinco partes básicas, como mostra a figura 5.5 [45].

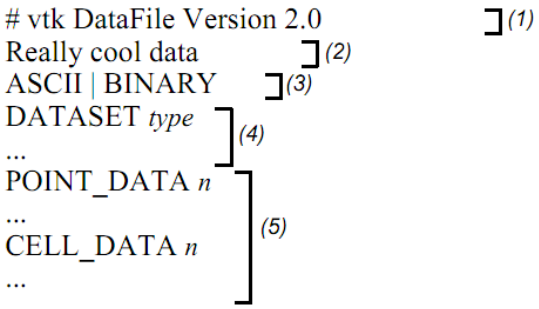


Fig. 5.5: Estrutura de um ficheiro *legacy* do VTK.

1. A primeira parte é a versão do ficheiro e identificador. Esta parte contém a seguinte linha, na qual "x.x"corresponde à versão do VTK:
   `# vtk DataFile Version x.x`.

2. A segunda parte é o cabeçalho que consiste numa *string* terminada pelo caracter de fim de linha, \n. O cabeçalho contém no máximo 256 caracteres e é usado para descrever os dados.

3. A terceira parte corresponde ao formato do ficheiro, podendo ser ASCII ou binário.

4. A quarta parte consiste na estrutura do conjunto de dados (*dataset*), e descreve a sua topologia e geometria. Esta parte inicia-se com uma linha contendo a palavra DATASET seguida de uma palavra indicando o tipo de estrutura, *structured_points*, *structured_grid*, *unstructured_grid*, *polydata*, *rectilinear_grid*, ou *field*. Para uma descrição detalhada dos tipos de estrutura de dados, aconselha-se leitura de [45].

5. A quinta parte descreve os atributos do *dataset*. Esta parte começa com as palavras POINT_DATA ou CELL_DATA, seguidas por um número inteiro especificando o número de pontos ou células, respectivamente. Outras combinações de palavras definem os valores dos atributos do *dataset* (i.e., escalares, vectores, normais, etc).

A conversão de VRML para vtkPolyData foi conseguida obtendo a geometria dos actores existentes no ficheiro VRML, e convertendo-a para vtkpolydata. A figura 5.6 apresenta um excerto de um ficheiro VTK originado no âmbito deste trabalho.

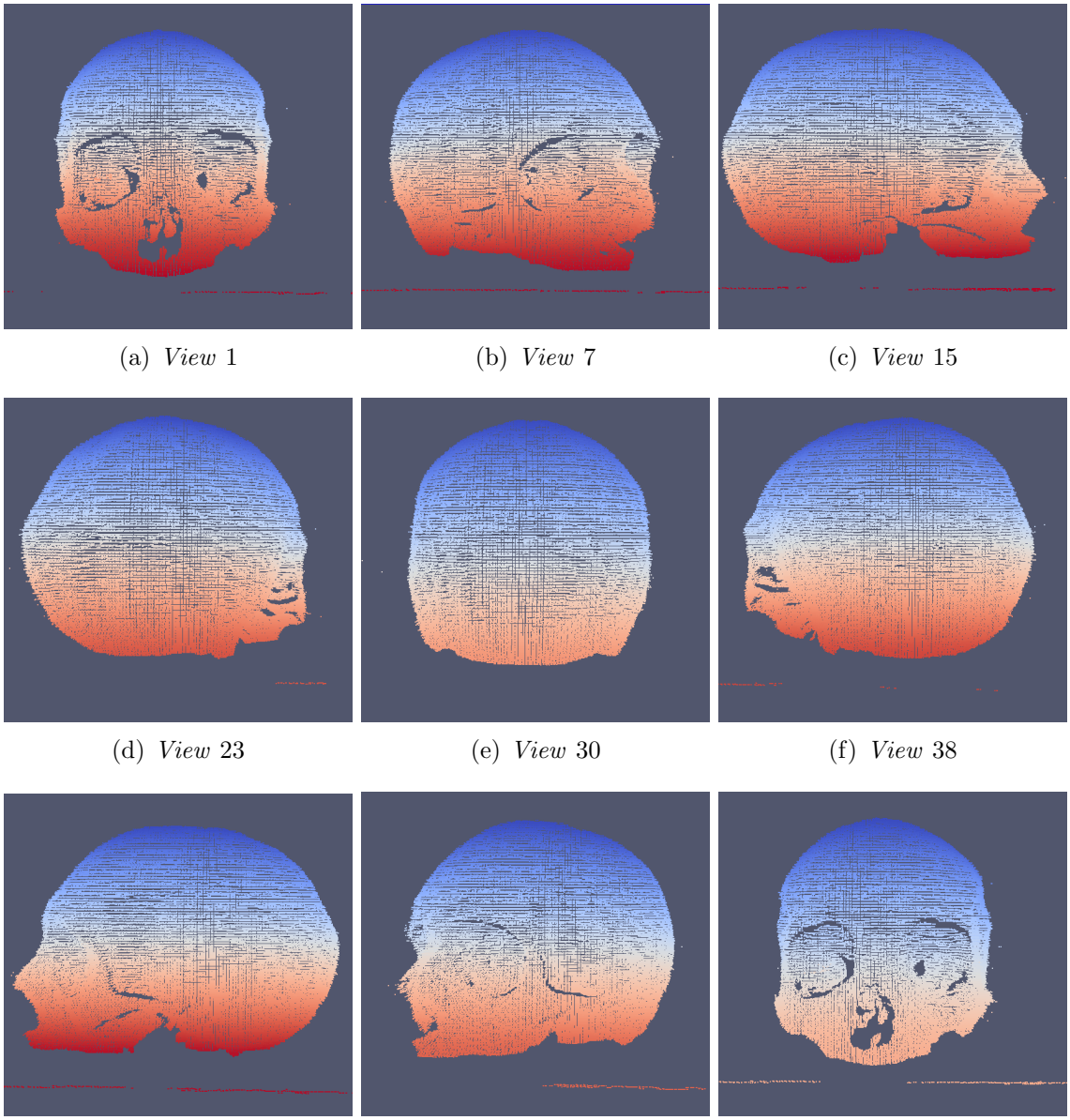
```
# vtk DataFile Version 3.0
vtk output
ASCII
DATASET POLYDATA
POINTS 77268 float
90.274 -58.4797 877.31
90.039 -58.4786 876.993
89.804 -58.4773 876.666
...
VERTICES 77268 154536
1 0
1 1
1 2
...
POINT_DATA 77268
SCALARS scalars float
LOOKUP_TABLE default
0 1 2 3 4 5 6 7 8
9 10 11 12 13 14 15 16 17
18 19 20 21 22 23 24 25 26
....
```

Fig. 5.6: Excerto de um ficheiro VTK originado.

Todas as nuvens de pontos adquiridas foram convertidas do formato VRML para o formato VTK.

5.5 Resultados e Discussão

Tal como indicado na secção 5.3, o *scanner* 3D forneceu um conjunto de 58 nuvens de pontos, ou seja 58 *views*, por cada *scan* realizado ao crânio. Cada *view* contém pontos da região da superífice do crânio visível pela câmara em cada aquisição. As regiões não visíveis pela câmara não foram adquiridas. Nas figuras 5.7 e 5.8 pode-se observar algumas das nuvens de pontos adquiridas nos *scans* 1 e 2 realizados.



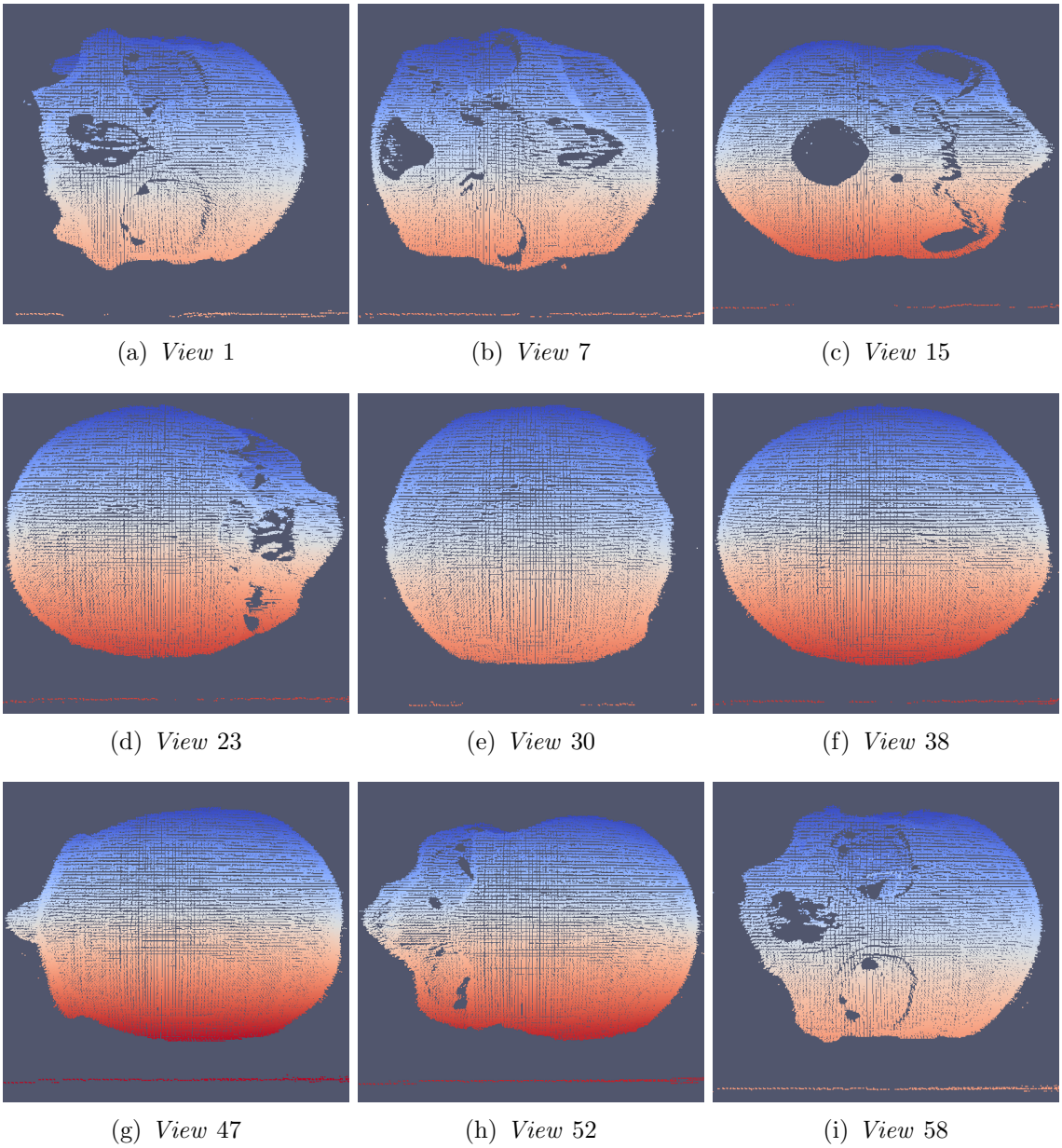
(a) *View* 1 (b) *View* 7 (c) *View* 15

(d) *View* 23 (e) *View* 30 (f) *View* 38

(g) *View* 47 (h) *View* 52 (i) *View* 58

Fig. 5.7: Nuvens de pontos obtidas no *scan* 1, correspondendo ao posicionamento vertical do crânio. O ângulo ϕ, entre um *view* e a posição inicial do crânio, pode ser calculado pela equação 4.1.

As coordenadas 3D dos pontos em cada nuvem correspondem a unidades físicas em milímetros (mm). Assim, por exemplo, um ponto com coordenadas $(0, 0, 750)$ segundo (x, y, z) encontra-se à distância de 750 mm da câmara na direcção definida pelo eixo *0z*. Devido aos planos de corte definidos no ficheiro "config.xml"(ver tabela 4.5), cada *view* possui pontos com coordenadas no intervalo definido entre 750 mm e 1000 mm segundo o eixo *0z*. A quantidade de pontos em cada *view* apresentado



(a) *View* 1 (b) *View* 7 (c) *View* 15

(d) *View* 23 (e) *View* 30 (f) *View* 38

(g) *View* 47 (h) *View* 52 (i) *View* 58

Fig. 5.8: Nuvens de pontos obtidas no *scan* 2, correspondendo ao posicionamento horizontal do crânio. O ângulo ϕ, entre um *view* e a posição inicial do crânio, pode ser calculado pela equação 4.1.

nas figuras 5.7 e 5.8 pode ser verificada na tabela 5.1. Através da análise da tabela 5.1 é possível verificar que a quantidade de pontos nos *views* varia entre 74000 e 110171. Este elevado número de pontos exige bastantes recursos computacionais e tempo para processar alguns algoritmos desenvolvidos.

Adicionalmente, além dos pontos correspondentes à superfície do crânio, está presente um conjunto de pontos ruidosos. Isto deve-se ao facto de os *scanners* 3D

Tab. 5.1: Número de pontos nos *views*, para algumas nuvens adquiridas.

N.º de pontos nas nuvens		
View	*Scan 1*	*Scan 2*
1	77268	74000
7	95887	80483
15	110171	91120
23	98830	95254
30	84588	84615
38	101745	96963
47	109621	102179
52	96177	99863
58	78113	77082

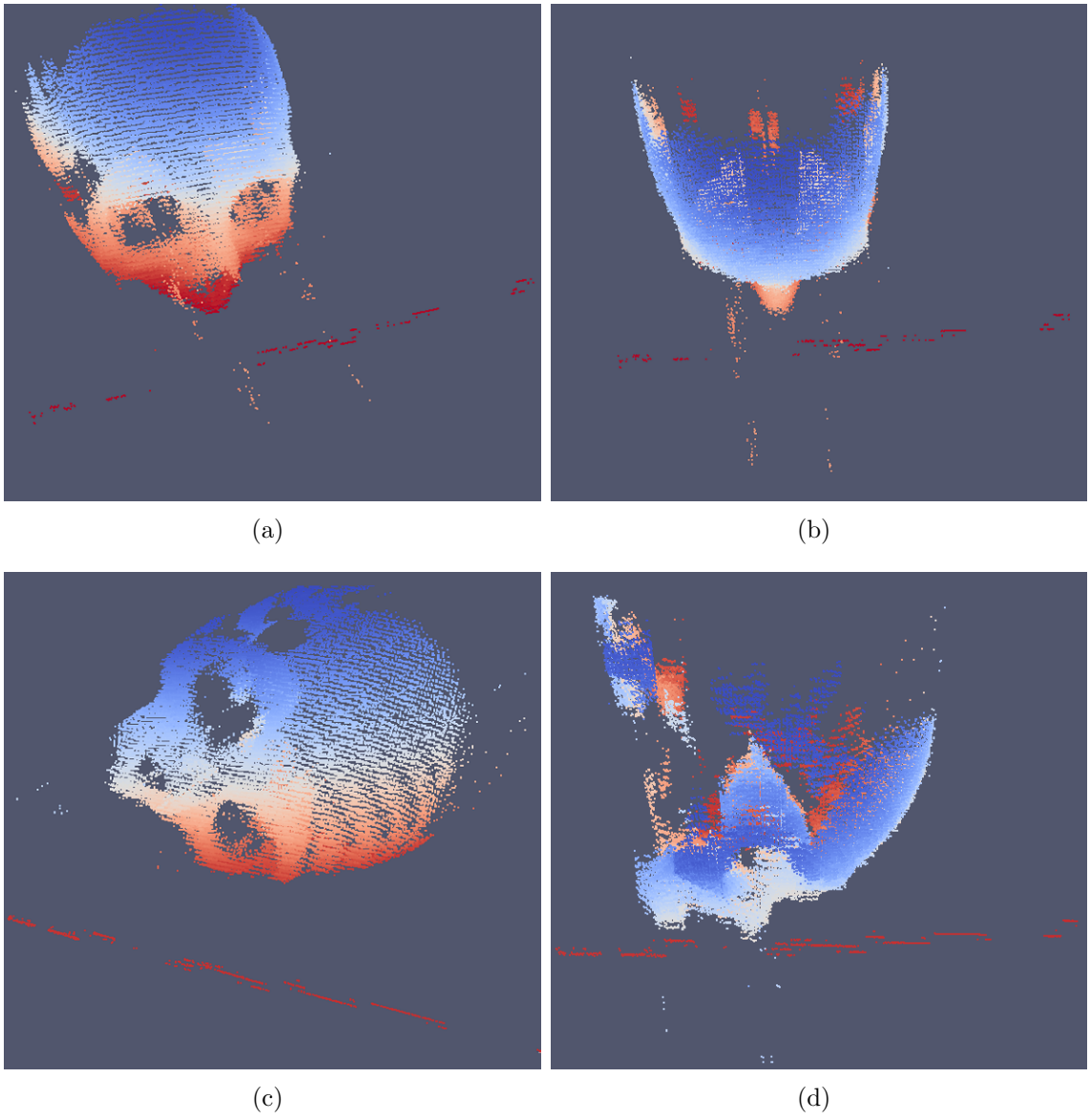
introduzirem erros na informação que adquirem. Existem pelo menos dois tipos de erros: erros de medição (aleatórios ou sistemáticos) das coordenadas dos pontos, e *outliers* que são pontos falsos e inconsistentes, apresentando um grande afastamento dos que efectivamente fazem parte da superfície analisada [46]. Estes pontos erróneos irão afectar a qualidade da superfície 3D final. Algumas possíveis fontes de introdução de erros são:

- Projector e/ou câmara desfocados;
- Luz espalhada a partir da superfície do objecto;
- Luz ambiente variável;
- Movimentação do objecto durante a projecção dos padrões.

A figura 5.9 apresenta o *view* 1 dos *scans* 1 e 2 sob diferentes orientações, evidenciando os *outliers* presentes nas nuvens de pontos.

5.6 Pré-Processamento

Considerando o elevado número de *outliers* e a quantidade de pontos nas nuvens realizou-se um pré-processamento das nuvens de pontos, que consistiu no desenvolvimento e na aplicação de dois algoritmos. O primeiro algoritmo remove *outliers* das nuvens de pontos, e o segundo algoritmo realiza uma sub-amostragem das nuvens de pontos. Nesta secção serão descritos dois algoritmos desenvolvidos para o pré-processamento das nuvens de pontos.



(a) (b)

(c) (d)

Fig. 5.9: *View* 1 sob diferentes orientações (à esquerda: vista oblíqua, à direita: vista superior), evidenciando os *outliers* presentes na nuvem de pontos. (a) e (b) *Scan* 1, (c) e (d) *Scan* 2.

5.6.1 Remoção de Ruído

Foi implementado, usando VTK, um algoritmo de pré-processamento para diminuir o número de pontos inconsistentes nas nuvens, tornando-as mais limpas e menos densas. Este algoritmo baseia-se na análise da distribuição das distâncias entre os pontos da nuvem. Para cada ponto, é calculada a distância média aos seus k vizinhos, ou seja, aos k pontos mais próximos. Seguidamente é calculada a média global das distâncias entre os pontos e os seus vizinhos, e determinado o desvio

padrão amostral. A partir daqui, define-se como inconsistente, qualquer ponto cuja distância média aos seus vizinhos seja superior em um desvio padrão (σ), ou a um múltiplo deste ($\alpha\sigma$), ao valor da distância média global. O algoritmo aceita como entrada nuvens do tipo vtkPolyData e produz à saída nuvens do mesmo tipo. Os parâmetros de entrada k e α são definidos pelo utilizador (figura 5.10).

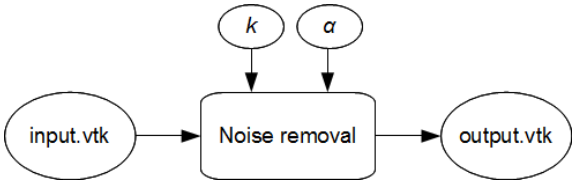


Fig. 5.10: Entradas e saídas do algoritmo para remoção de *outliers* de uma nuvem de pontos.

O algoritmo lê as nuvens de pontos através da classe vtkPolyDataReader, fornecendo o *input* da classe vtkPolyData. Considerando então uma nuvem de pontos de entrada com N pontos $\{p_1, ..., p_N\}$ de coordenadas (x, y, z), para cada ponto P $\in \{p_1, ..., p_N\}$ vão ser encontrados os seus k vizinhos. Para tal, é criada uma *K-D Tree* recorrendo à classe vtkKdTreePointLocator e ao método FindClosestNPoints. Os pontos vizinhos são armazenados num *array* do tipo vtkIdList, que é acedido para o cálculo da distância média entre P e estes, (d_P).

$$d_P = \frac{1}{k}\sum_{j=1}^{k}\sqrt{(P_x - P_{j_x})^2 + (P_y - P_{j_y})^2 + (P_z - P_{j_z})^2}, \forall P \in \{P_1, ..., P_N\}, P \neq P_j \tag{5.3}$$

A cada ponto P é associado o respectivo valor de d_P. De seguida é calculada a distância média global entre os pontos e seus vizinhos (d_g), através do cálculo do valor médio de d_P.

$$d_g = \frac{1}{N}\sum_{i=1}^{N} d_{P_i} \tag{5.4}$$

Procede-se para o cálculo do desvio padrão da distância média entre os pontos e respectivos vizinhos.

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(d_{P_i} - d_g)^2} \tag{5.5}$$

A partir daqui, verificam-se os valores de d_P para cada ponto P, e os IDs de todos os pontos com valores dentro do intervalo fechado $[d_g; d_g + \alpha\sigma]$, são introduzidos num

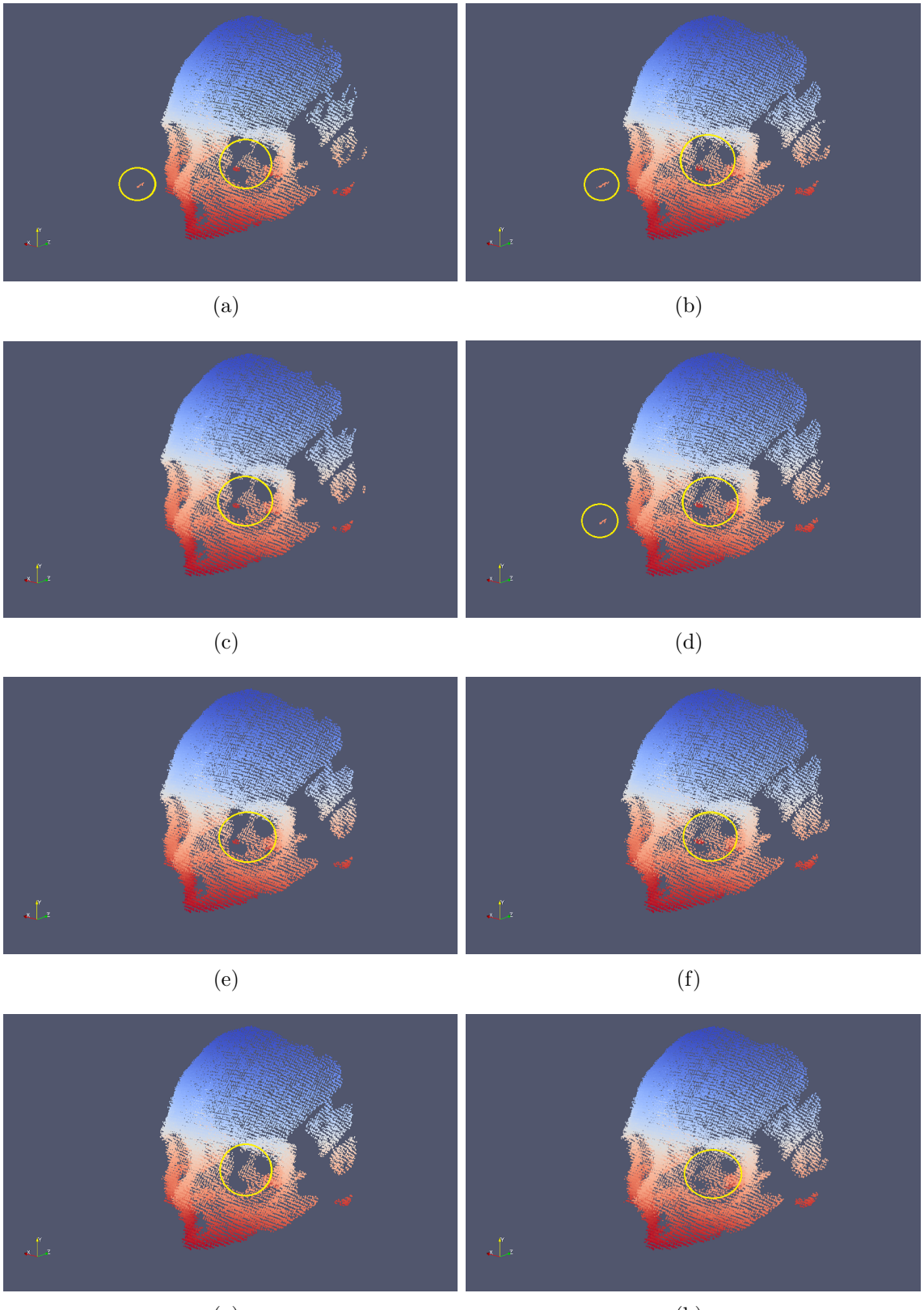
array do tipo vtkIdTypeArray. Através das classes vtkSelectionNode, vtkSelection e vtkExtractSelection, esses pontos são selecionados e extraídos, originando uma grelha não estruturada do tipo vtkUnstructuredGrid. A classe vtkDataSetSurfaceFilter é então utilizada para converter o resultado final numa nova nuvem do tipo vtkPolyData. Para se verificar a influência dos parâmetros de entrada, k e α, no resultado final, foram realizados alguns testes no *view* 1 de cada *Scan* realizado. Os parâmetros dos testes e a sua influência no cálculo de d_g, de σ e no número de pontos removidos são descriminados na tabela 5.2. As figuras 5.11 e 5.12 apresentam os resultados dos testes realizados, salientando as principais diferenças na remoção de pontos.

Com base nos testes realizados verifica-se que, aumentando o parâmetro k diminui-se o número de pontos removidos, mas verifica-se que os pontos são removidos de forma mais assertiva, tendendo a fazer desaparecer as "ilhas de pontos". No entanto, gastam-se mais recursos computacionais, e o tempo de processamento é maior. Uma alternativa consiste em diminuir o valor do parâmetro α de forma a obter um processamento mais rápido. No entanto, verifica-se a remoção de muitos pontos que não constituem *outliers*, o que pode comprometer informação importante contida na nuvem. Com base no conjunto de testes realizado, verifica-se que com os valores k=100, α=1 é atingido um bom compromisso entre tempo de processamento das nuvens de pontos e a qualidade do resultado final.

A qualidade do resultado final é tanto melhor quanto mais *outliers* forem removidos, e mais pontos da superfície do crânio forem preservados. Em suma, considerando os resultados obtidos com a implementação deste algoritmo, pode afirmar-se que este constitui um método simples e eficaz para a removação de *outliers* numa nuvem de pontos.

Tab. 5.2: Testes com o algoritmo de remoção de ruído.

k	α	*Scan* 1, *View* 1				*Scan* 2, *View* 1			
		d_g	σ	Orig.	Rem.	d_g	σ	Orig.	Rem.
25	0.5	1.702	0.7231	77268	9667	1.705	0.801	74000	8441
25	1	1.702	0.7231		3878	1.705	0.801		3139
50	0.5	2.311	1.174		6487	2.311	1.049		7370
50	1	2.311	1.174		2928	2.311	1.049		3389
100	0.5	3.076	2.140		5097	3.065	1.569		6963
100	1	3.076	2.140		2126	3.065	1.569		3373
150	0.5	3.652	2.810		4812	3.645	2.088		6491
150	1	3.652	2.810		2007	3.645	2.088		3155



(a) (b) (c) (d) (e) (f) (g) (h)

Fig. 5.11: Resultados do algoritmo de remoção de ruído para o *view* 1, *scan* 1: (a) k=25, α=0.5 (b) k=25, α=1 (c) k=50, α=0.5 (d) k=50, α=1 (e) k=100, α=0.5 (f) k=100, α=1 (g) k=150, α=0.5 (h) k=150, α=1. As principais diferenças estão assinaladas a amarelo.

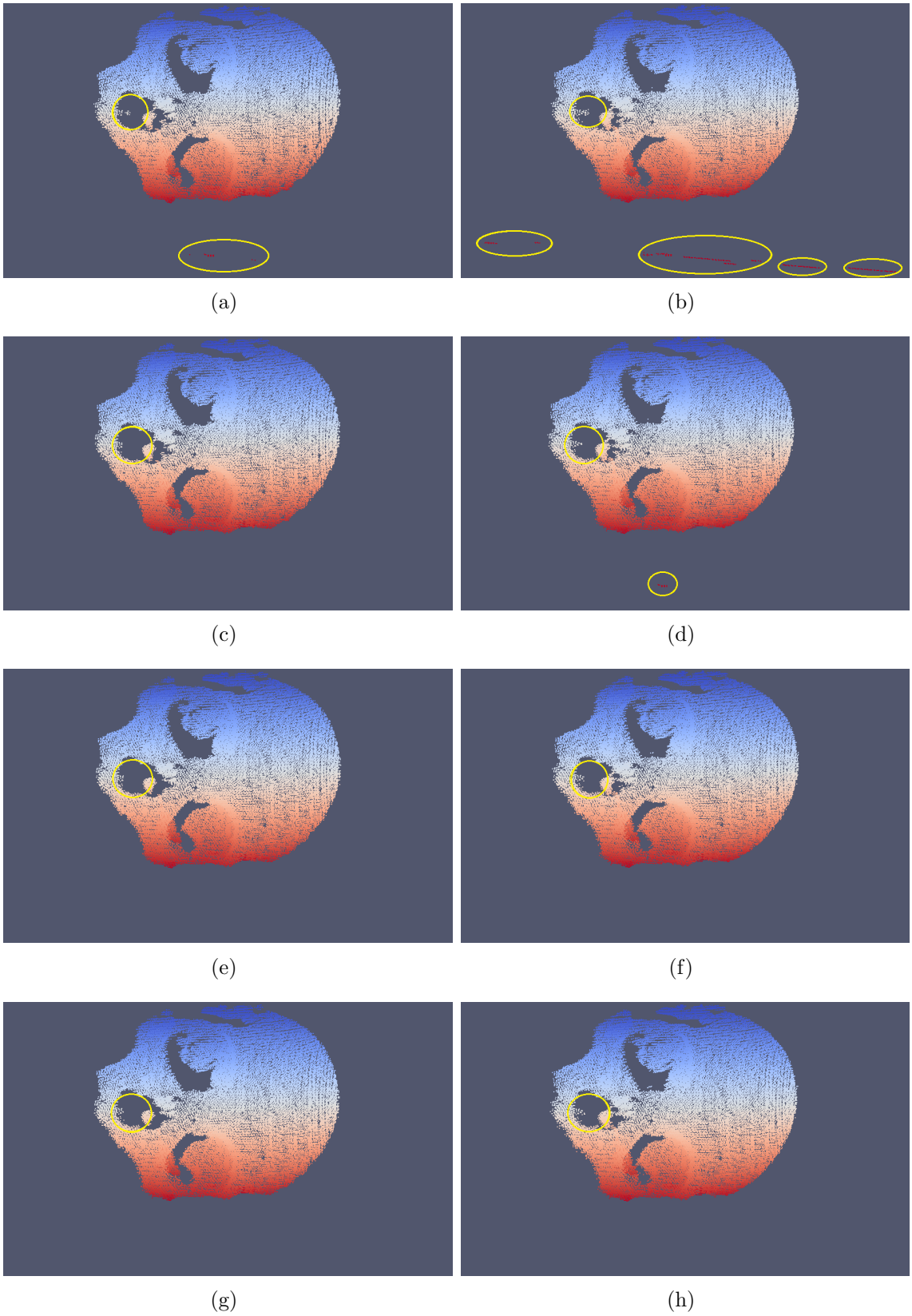


Fig. 5.12: Resultados do algoritmo de remoção de ruído para o *view* 1, *scan* 2: (a) k=25, α=0.5 (b) k=25, α=1 (c) k=50, α=0.5 (d) k=50, α=1 (e) k=100, α=0.5 (f) k=100, α=1 (g) k=150, α=0.5 (h) k=150, α=1. As principais diferenças estão assinaladas a amarelo.

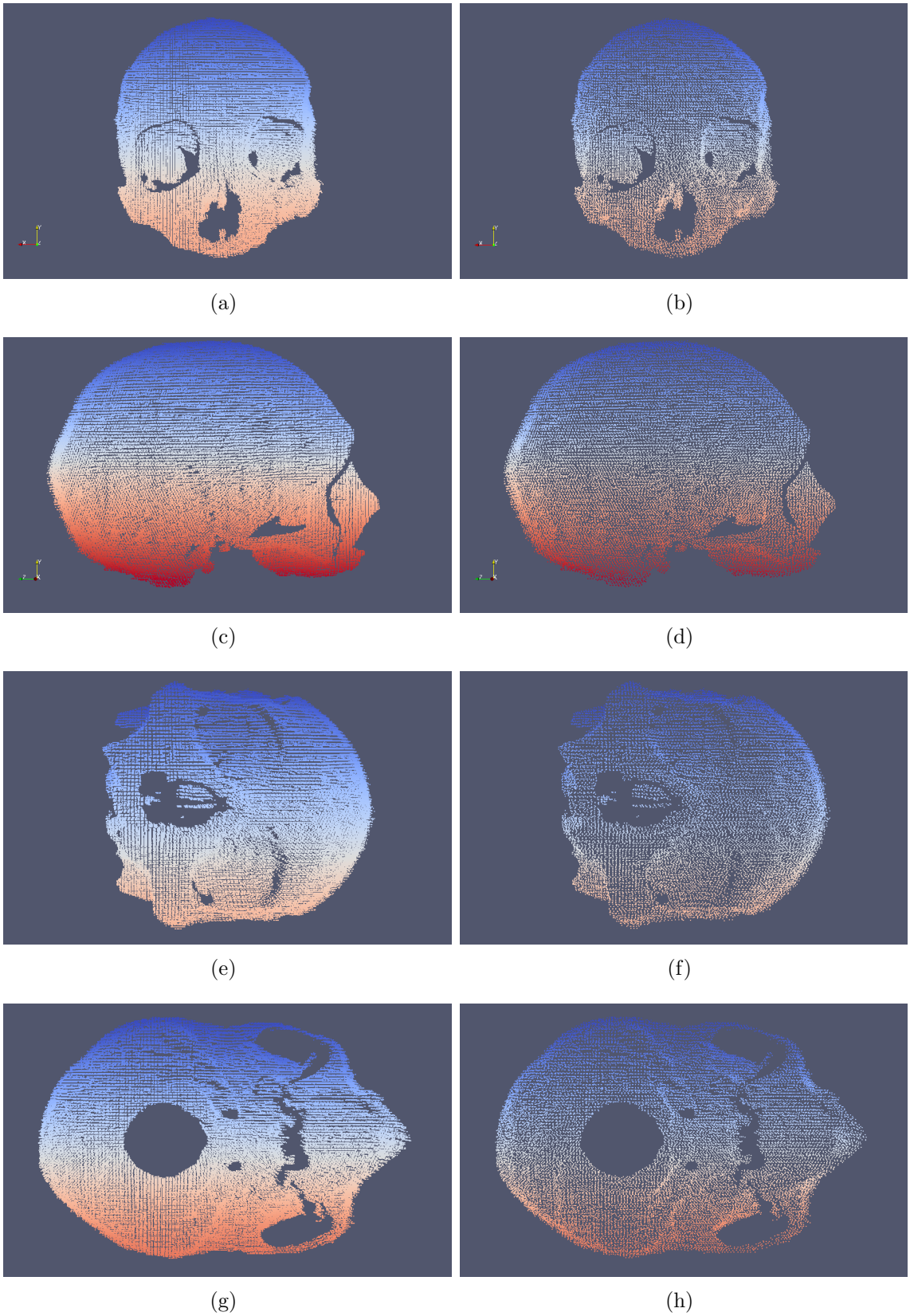
5.6.2 Sub-Amostragem

O algoritmo para sub-amostragem das nuvens de pontos foi implementado usando VTK, e foi designado de *Downsample*. O objectivo da sua implementação passou por diminuir a densidade de pontos nas nuvens mas preservar ao máximo a informação sobre a estrutura representada. Este algoritmo foi aplicado recorrendo à classe vtkCleanPolyData que diminui o número de pontos nas nuvens através da fusão de pontos coincidentes e da conversão de células degenerativas em formas apropriadas (polígonos em linhas, linhas em pontos; por exemplo, um triângulo é convertido numa linha caso dois dos seus vértices sejam fundidos) [45]. Este filtro aceita como parâmetro de entrada um valor de tolerância (*Tolerance*), que corresponde a uma fracção do comprimento da *bounding-box* de cada nuvem, e que determina a distância mínima a que dois pontos podem estar um do outro. Após um conjunto de testes, o parâmetro de tolerância foi definido como 0.005, podendo-se verificar na tabela 5.3 o efeito do filtro *Downsample* na quantidade de pontos em alguns *views* dos *scans* 1 e 2, a título de exemplo.

Tab. 5.3: Número de pontos nas nuvens originais e sub-amostradas com o algoritmo *Downsample*.

Tolerance	*View*	*Scan* 1		*Scan* 2	
		Original	*Downsample*	Original	*Downsample*
0.005	1	75142	21463	70627	18447
	7	92201	25743	76098	21081
	15	106520	26822	87568	23185
	23	95052	24164	89691	21839
	30	81874	25421	83182	24775
	38	99628	24873	92706	26225
	47	106100	25694	97237	23486
	52	93394	25073	94776	23975
	58	75231	21697	72643	20591

Na figura figura 5.13 é possível ter uma percepção visual do efeito do algoritmo *Downsample* em duas nuvens de pontos diferentes (*view* 1 e *view* 15) dos *scans* 1 e 2. Em ambas as nuvens a estrutura do crânio foi preservada, mas a densidade de pontos é inferior à das nuvens originais. A tabela 5.4 apresenta a percentagem de redução do número de pontos nos diferentes *views*.



(a) (b) (c) (d) (e) (f) (g) (h)

Fig. 5.13: Sub-amostragem de diferentes nuvens de pontos. (a) e (b) *Scan* 1, *view* 1 original (75142 pontos) e *downsampled* (21463 pontos). (c) e (d) *Scan* 1, *view* 15 original (106258 pontos) e *downsampled* (26427 pontos). (e) e (f) *Scan* 2, *view* 1 original (70627 pontos) e *downsampled* (18447 pontos). (g) e (h) *Scan* 2, *view* 15 original (87568 pontos) e *downsampled* (23185 pontos).

Tab. 5.4: Percentagem de redução do número de pontos nas nuvens com o algoritmo *Downsample*.

Tolerance	*View*	% de redução	
		Scan 1	*Scan* 2
0.005	1	71.44	73.88
	7	72.08	72.3
	15	74.82	73.52
	23	74.58	75.65
	30	68.95	70.22
	38	75.03	71.71
	47	75.78	75.85
	52	73.15	74.7
	58	71.16	71.65

Graças ao algoritmo de sub-amostragem -*Downsample*-, são necessários menos recursos computacionais e menos tempo para processar as nuvens de pontos. O filtro *Downsample* foi aplicado a todos os *views*, sendo no entanto preservadas as nuvens originais.

Capítulo 6

Reconstrução de Superfícies

Neste capítulo será abordada a metodologia utilizada para a reconstrução de superfícies a partir das nuvens de pontos obtidas no capítulo anterior. Será também descrito o método implementado para a validação das superfícies geradas, fundamental para se quantificar a qualidade das mesmas.

6.1 O Problema da Reconstrução de Superfícies

A reconstrução de superfícies a partir de nuvens de pontos obtidas por um *scanner* 3D constitui ainda hoje um problema em aberto, existindo diferentes métodos para originar modelos poligonais a partir de conjuntos de pontos [47]. Em alguns trabalhos [1, 9, 23, 47] é possível identificar diferentes abordagens à reconstrução de superfícies. No entanto, existe um conjunto de passos transversal às várias abordagens, composto por:

- Pré-processamento das nuvens de pontos (remoção de ruído, sub-amostragem, etc);
- Alinhamento das nuvens de pontos segundo um sistema de coordenadas comum;
- Integração das nuvens de pontos numa só nuvem global;
- Construção de um modelo geométrico poligonal que represente a superfície completa do objecto em estudo.

O pré-processamento das nuvens de pontos, realizado e descrito na secção 5.6, torna as nuvens menos redundantes e menos densas, diminuindo significativamente

o número de pontos inconsistentes. Este facto traduz-se numa redução do tempo de computação necessário para correr alguns algoritmos e numa qualidade superior do modelo final.

O alinhamento das nuvens de pontos é realizado em relação a um sistema de coordenadas comum a todas as nuvens. Quando a posição e a orientação do objecto face ao *scanner* 3D são alteradas de forma controlada (por uma plataforma rotativa, por exemplo), é possível definir uma transformação que relacione as diferentes posições e orientações que o objecto assume. Consequentemente, as nuvens de pontos do objecto vão estar relacionadas pela mesma trasformação, que pode ser aplicada computacionalmente para alinhar as nuvens. Quando esta transformação não é conhecida, recorre-se a algoritmos que estimam uma transformação que sobreponha ao máximo, os pontos das diferentes nuvens. Também é possível alinhar as nuvens de pontos através da combinação destas duas formas. As nuvens de pontos podem ser alinhadas duas a duas (*Pairwise Registration*), ou todas em simultâneo (*Parallel Registration*) [1]. A vantagem do alinhamento aos pares é o seu baixo custo computacional, por outro lado, os erros de cada alinhamento são adicionados aos erros do alinhamento anterior, tal que o resultado final apresenta uma acumulação de erros. O alinhamento em simultâno evita a acumulação de erros ao alinhar todas as nuvens ao mesmo tempo, dispersando os erros pelas diferentes nuvens. No entanto, o alinhamento em simultâneo exige recursos computacionais bastante superiores comparativamente com o alinhamento aos pares [1]. Uma vez que cada nuvem de pontos contém informação sobre uma pequena porção da superfície do objecto em estudo, para se produzir uma representação completa da sua superfície, é necessário integrar os pontos das diferentes nuvens, originando uma nuvem de pontos global. Esta integração das nuvens de pontos é linear e simples. Quando as nuvens de pontos são alinhadas aos pares, é fixada uma nuvem como referência. As nuvens de pontos são alinhadas com a nuvem de referência, que recebe os seus pontos no final de cada alinhamento. Desta forma, a nuvem de referência vai tendo cada vez mais pontos, e cada nuvem é alinhada com o conjunto de todos os pontos já alinhados. Quando as nuvens de pontos são alinhadas simultaneamente, a sua integração também é simultânea.

Por fim, a reconstrução de um modelo poligonal que represente uma superfície é um tema actualmente bastante explorado, não existindo um método definido como padrão. A reconstrução pode ser feita a partir de conjuntos de pontos não organizados (só coordenadas x, y, z dos pontos), ou a partir de dados estruturados (como uma imagem ou uma grelha estruturada). Em [9] (pp. 78-83), é possível consul-

tar alguns métodos para gerar uma superfície, a partir de conjuntos de informação diferentes.

Nas secções seguintes serão apresentados e descritos os algoritmos implementados para cada um dos passos do processo de reconstrução de superfícies, com excepção do pré-processamento das nuvens, já descrito na secção 5.6. Será também apresentado e descrito o método utilizado para validação das superfícies obtidas.

6.2 Alinhamento e Integração de Nuvens de Pontos

Para o alinhamento e integração das nuvens de pontos foi desenvolvido um algoritmo utilizando VTK e ITK. Este algoritmo, designado de *Alinhamento e Integração* (AI), alinha várias nuvens de pontos (*views*), integrando os seus pontos numa nuvem global e mapeando-os numa imagem 3D. A figura 6.1 apresenta as entradas e saídas do algoritmo AI.

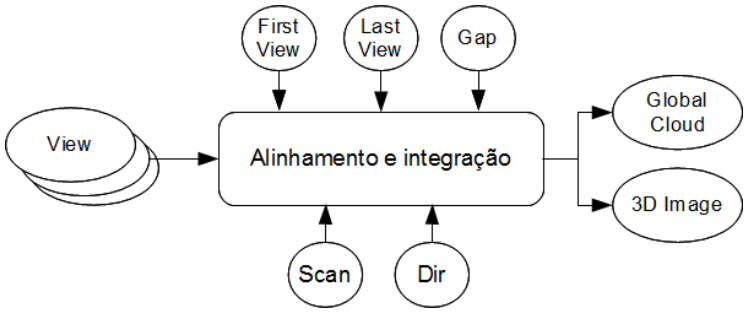


Fig. 6.1: Entradas e saídas do algoritmo de Alinhamento e Integração.

Uma vez que o *scanner* 3D pode realizar vários *scans* a um dado objecto, adquirindo em cada um, vários *views* (ver figura 4.6, secção 4.3.3), ao usar o algoritmo AI é necessário indicar qual o *scan* pretendido e, dentro do *scan*, quais os *views* a alinhar e integrar. São solicitados os seguintes parâmetros ao utilizador:

- *Dir* - O directório onde se encontram as nuvens de pontos;
- *Scan* - O conjunto de *views* a ser lido;
- *First View* - Primeiro *view* a ser lido;
- *Last View* - Último *view* a ser lido;
- *Gap* - O intervalo de leitura entre *views*.

Assim, por exemplo, para determinado objecto, o utilizador pode seleccionar o *scan* 2 e os *views* 2 (*First View*) a 58 (*Last View*), com um intervalo de leitura (*Gap*), de 3 *views.* Ou seja, serão lidos os *views* 2, 5, 8, 11,..., até ao *view* 58. À saída, a nuvem de pontos global é do tipo vtkPolyData, e a imagem 3D é do tipo itk::Image. A figura 6.2 apresenta a *pipeline* do algoritmo AI, indicando os principais parâmetros de entrada de cada filtro utilizado. O algoritmo AI é composto por três passos fundamentais, que são:

- Correcção da orientação das nuvens de pontos;
- Alinhamento e integração das nuvens de pontos;
- *Splatting* dos pontos numa imagem 3D.

Estes três passos principais (Correcção da orientação/ Alinhamento e integração/ *Splatting*) encontram-se destacados na *pipeline* do algoritmo AI (figura 6.2) a cinzento, verde e laranja, respectivamente. Segue-se uma explicação de cada um dos passos principais do algoritmo AI.

6.2.1 Correcção da Orientação

Devido à rotação introduzida no objecto de estudo entre aquisições de pontos, as nuvens de pontos apresentam orientações diferentes face ao sistema de coordenadas comum, que é o da câmara (secção 4.3.3). Cada nuvem de pontos (*view*), apresenta uma rotação ϕ, em relação à posição incial do objecto, dada pela equação 4.1.

Para se corrigir a orientação das nuvens de pontos foi implementado um filtro que se designou *Rigid Transform.* Este filtro recebe uma tranformação rígida, constituída por uma rotação e por uma translação, e aplica-a a uma nuvem de pontos. O filtro e a transformação rígida são definidos recorrendo às classes vtkTransformPolyDataFilter e vtkTransform. Assim, para dar a mesma orientação a todas as nuvens de pontos foi definida uma transformação rígida designada *TF0.* Com esta transformação, definida por uma translação e por uma rotação, cada nuvem de pontos é trazida para a origem do sistema de coordenadas e sofre uma rotação no sentido contrário ao aplicado pela plataforma rotativa.

É importante referir que, ao longo das aquisições de pontos, a posição do objecto em estudo é considerada constante relativamente à câmara, sendo alterada somente a sua orientação devido à rotação. A rotação do objecto dá-se em torno do eixo da plataforma rotativa, considerado como um eixo paralelo ao eixo $0y$ do centro

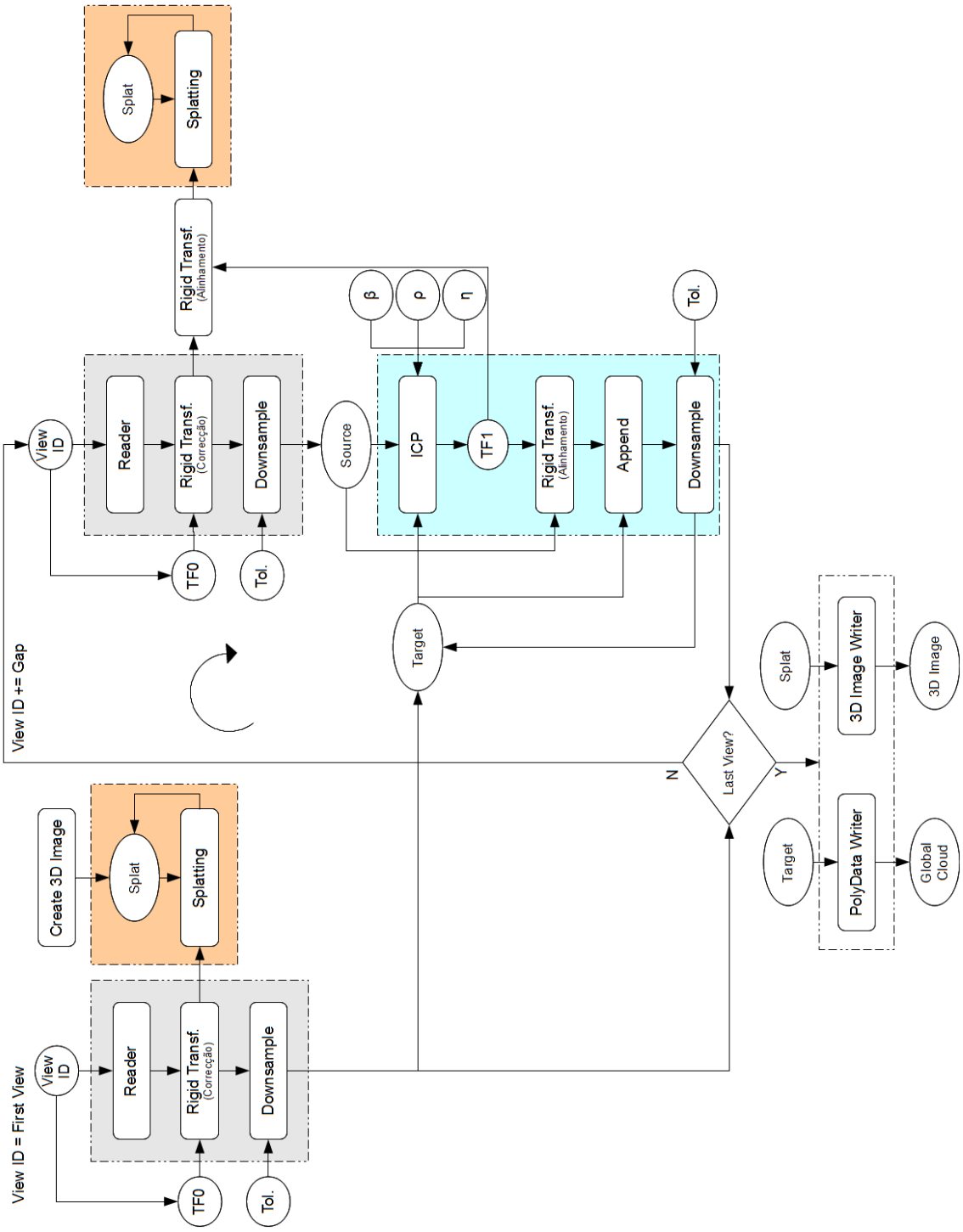


Fig. 6.2: *Pipeline* do algoritmo de Alinhamento e Integração.

de coordenadas da câmara. Devido à separação física entre o centro da plataforma rotativa e a câmara, e tendo em conta a orientação da câmara, as coordenadas do centro da plataforma rotativa são (70, -80, 875) mm segundo (x, y, z). Assim,

para posicionar a região do centro da plataforma rotativa na origem do sistema de coordenadas foi necessário aplicar uma translação $\mathbf{T}$ composta por uma deslocação segundo cada eixo, dada por $\mathbf{T}_x = -70$, $\mathbf{T}_y = 80$, e $\mathbf{T}_z = -875$. A posição final p' de um ponto p correspondente ao centro da plataforma rotativa é dada por:

$$\begin{bmatrix} p'_x & p'_y & p'_z & 1 \end{bmatrix} = \begin{bmatrix} p_x & p_y & p_z & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \mathbf{T}_x & \mathbf{T}_y & \mathbf{T}_z & 1 \end{bmatrix}$$

Resolvendo o sistema de equações lineares tem-se:

$$p'(x,y,z) = \begin{cases} p'_x = p_x + \mathbf{T}_x \\ p'_y = p_y + \mathbf{T}_y \\ p'_z = p_z + \mathbf{T}_z \end{cases} = \begin{cases} p'_x = 70 + (-70) \\ p'_y = -80 + 80 \\ p'_z = 875 + (-875) \end{cases} \begin{cases} p'_x = 0 \\ p'_y = 0 \\ p'_z = 0 \end{cases} \tag{6.1}$$

Com a nuvem de pontos centrada na origem do sistema de coordenadas é possível compensar a rotação ϕ, aplicando uma rotação igual no sentido inverso ($\alpha = -\phi$), segundo o eixo $0y$ dada por:

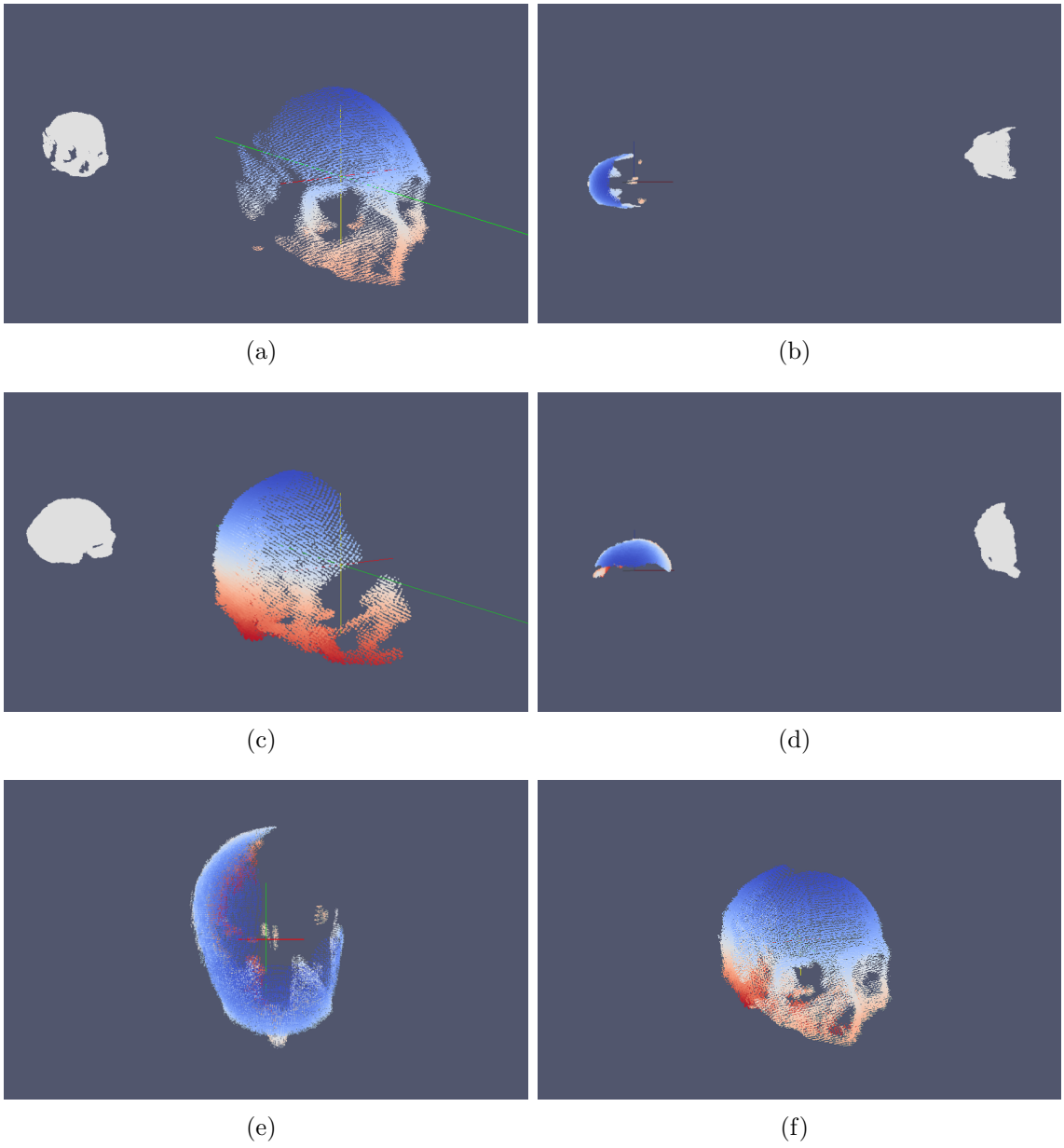
$$\begin{bmatrix} p'_x & p'_y & p'_z & 1 \end{bmatrix} = \begin{bmatrix} p_x & p_y & p_z & 1 \end{bmatrix} \times \begin{bmatrix} \cos\alpha & 0 & -sen\alpha & 0 \\ 0 & 1 & 0 & 0 \\ sen\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Resolvendo o sistema de equações lineares tem-se:

$$p'(x,y,z) = \begin{cases} p'_x = p_x \times \cos\alpha + p_z \times sen\alpha \\ p'_y = p_y \\ p'_z = p_z \times \cos\alpha - p_x \times sen\alpha \end{cases} \tag{6.2}$$

A translação $\mathbf{T}$ é comum a todas as nuvens de pontos, uma vez que a posição do objecto face à câmara é fixa, enquanto que a rotação α é diferente para cada nuvem. As figuras 6.3 e 6.4 apresentam o resultado da transformação rígida *TF0* nas nuvens de pontos *view* 1 e *view* 15 dos *scans* 1 e 2, a título de exemplo.

Após a aplicação da transformação rígida *TF0*, as nuvens de pontos apresentam um bom alinhamento comparativamente com as suas posições originais. No entanto, existe um desfasamento perceptível entre os pontos das diferentes nuvens. Este desfasamento deve-se a alguns movimentos não contemplados do crânio sobre a plataforma, e ao posicionamento e orientação exactos da câmara face à plataforma



(a) (b)

(c) (d)

(e) (f)

Fig. 6.3: Transformação rígida, *TF0*, em diferentes *views* do *scan* 1, com a posição original das nuvens a cinzento. (a) e (b) *View* 1 (c) e (d) *View* 15 (e) e (f) *Views* 1 e 15 após aplicação de *TF0*.

rotativa.

Após a correcção da orientação, cada nuvem de pontos é submetida ao filtro de sub-amostragem *Downsample* (ver secção 5.6.2). O objectivo é permitir o processamento de nuvens de pontos mais densas, que não tenham sido sub-amostradas previamente. No caso da leitura de nuvens já sub-amostradas pelo filtro *Downsample*, o número de pontos não é reduzido novamente, uma vez que o parâmetro

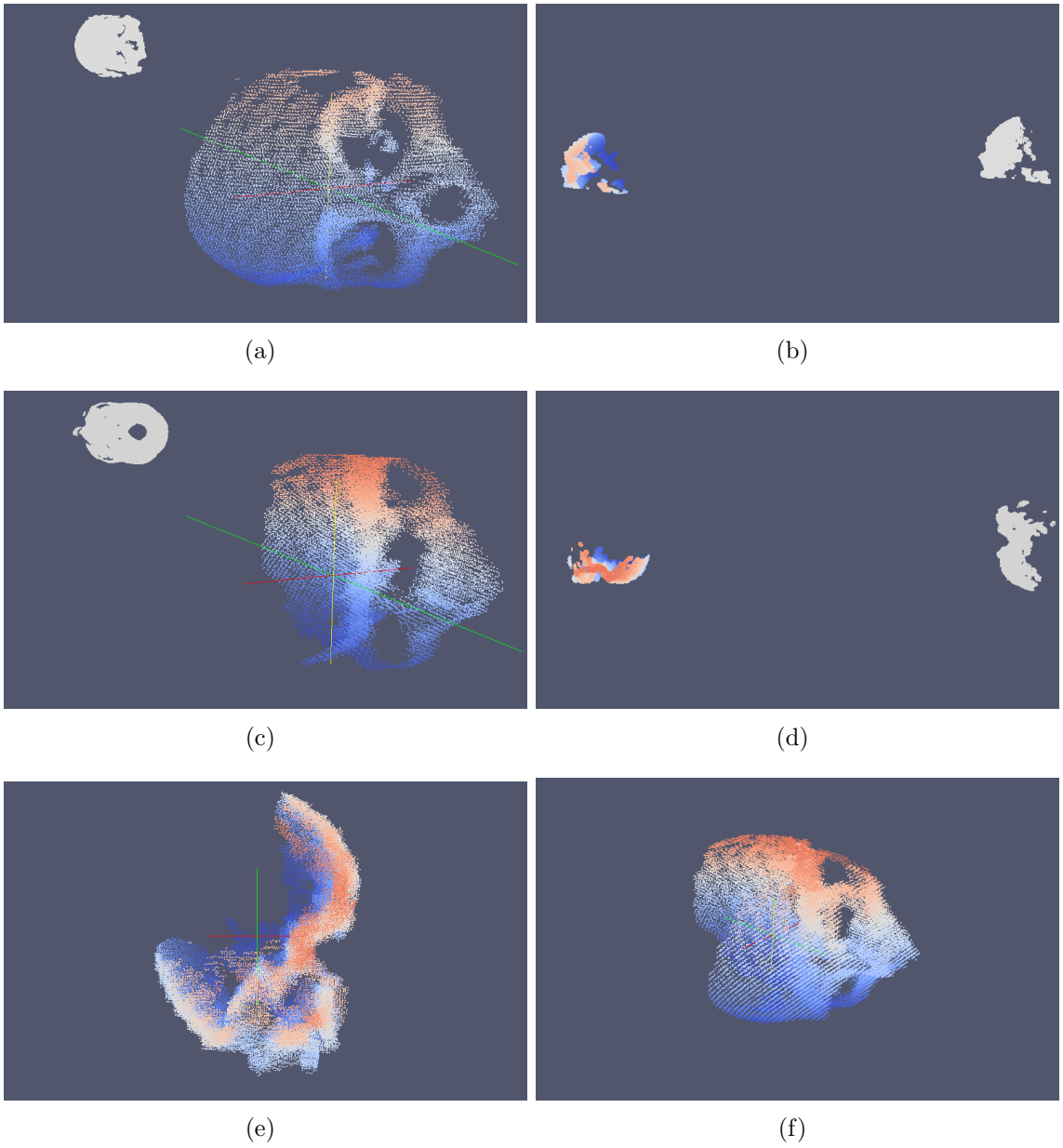


Fig. 6.4: Transformação rígida, *TF0*, em diferentes *views* do *scan* 2, com a posição original das nuvens a cinzento. (a) e (b) *View* 1 (c) e (d) *View* 15 (e) e (f) *Views* 1 e 15 após aplicação de *TF0*.

de tolerância é inalterado (*Tolerance* = 0.005). As nuvens de pontos devidamente orientadas e não sub-amostradas são preservadas.

6.2.2 Alinhamento e Integração

Uma vez que, após a correcção da orientação, as nuvens de pontos (*views*), não apresentam o melhor alinhamento, é necessário determinar uma transformação rígida que melhor alinhe as diferentes nuvens de pontos. Adicionalmente, como cada nuvem de pontos contém informação de uma pequena porção da superfície do objecto, é necessário integrar as diferentes nuvens para obter uma representação completa da sua superfície. Para o alinhamento das nuvens de pontos recorreu-se ao algoritmo *Iterative Closest Point* (ICP). O algoritmo ICP foi desenvolvido independentemente por um conjunto de autores como Besl e McKay [48] e Zhang [49], tendo sofrido várias alterações, o que levou ao aparecimento de variantes do algoritmo inicial, algumas das quais podem ser consultadas em [50, 51]. O ICP estima uma transformação rígida, composta por uma matriz de rotação $\mathbf{R} \in \mathbb{R}^{3\times3}$, e por um vector de translação $\mathbf{T} \in \mathbb{R}^3$, que permita atingir a máxima sobreposição de pontos correspondentes em dois conjuntos de pontos diferentes. Considerando duas nuvens de pontos, $\{p_1, ..., p_N\}$ (*source*), e $\{q_1, ..., q_M\}$ (*target*), o ICP vai procurar iterativamente a transformação rígida que minimize a função de custo, dada pela média do erro quadrático entre pontos correspondentes nas duas nuvens a alinhar. Essa função é dada por:

$$\phi(\mathbf{R}, \mathbf{T}) = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{R}p_i + \mathbf{T} - q_i\|^2 \tag{6.3}$$

A procura pela transformação termina quando a distância média entre pontos correspondentes nas duas nuvens é inferior a um valor específicado, ou quando é atingido o número limite de iterações definido pelo utilizador. Um dos problemas do ICP é necessitar de um bom posicionamento inicial das nuvens a alinhar. Graças à transformação *TF0* aplicada previamente (secção 6.2.1), tem-se um bom ponto de partida para o ICP estimar uma nova transformação *TF1*, para a máxima convergência dos pontos correspondentes nas nuvens. O ICP aceita diversos parâmetros de entrada, sendo os principais:

- *Source*, que corresponde à nuvem de pontos a alinhar;
- *Target*, que é a nuvem de pontos de referência;
- β, para o número máximo de iterações;
- ρ, para o número máximo de marcadores (*Landmarks*);
- η, para o parâmetro *Too far Threshold*.

A nuvem de referência - *Target* -, é fixa e corresponde à primeira nuvem de pontos (*First View*), corrigida e sub-amostrada. A nuvem a alinhar - *Source* -, é variável e corresponde ao *view* lido na execução do *loop* em questão, corrigido e sub-amostrado. Os parâmetros β e ρ definem critérios de paragem do algoritmo, sendo que β indica o número máximo de iterações para a convergência das nuvens, e ρ indica o número máximo de pontos de referência nas nuvens de pontos. Uma vez que as nuvens de pontos são densas, não é necessário utilizar todos os seus pontos, sendo suficiente recorrer-se a um subconjunto (*landmarks*). Após um conjunto de testes para tentar encontrar a melhor combinação dos parâmetros, os valores definidos foram $\beta = 100$, $\rho = 2000$ e $\eta = 20$ mm. O VTK disponibiliza a versão mais básica do *Iterative Closest Point*. Para cada ponto na nuvem *Target*, o algoritmo procura pelo ponto mais próximo na nuvem *Source* e estima a transformação de forma a minimizar a distância entre pontos correspondentes. Desta forma, pontos de todas as regiões nas nuvens vão ser contabilizados na computação da transformação. Se as nuvens *Target* e *Source* representarem diferentes regiões do objecto de estudo, e o algoritmo contabilizar os pontos fora da região de sobreposição das nuvens, o alinhamento vai tender a aumentar a sobreposição entre elas. Como resultado, a transformação vai alinhar as nuvens incorrectamente. Uma solução para este problema consiste em considerar só os pontos na região de sobreposição das nuvens *Source* e *Target* para a computação da transformação. Assim, se um ponto na nuvem *Source* não tiver um ponto correspondente na nuvem *Target* dentro de uma determinada distância η, assume-se que não consta da região de sobreposição e não é considerado na computação da transformação. Esta distância η é o parâmetro de entrada do método *SetTooFarThreshold* definido por Luca Antiga e David Doria, ambos desenvolvedores activos da comunidade VTK. Este método faz com que o algoritmo ICP não considere os pontos fora da região de sobreposição das nuvens para o cálculo da transformação. A figura 6.5 evidencia o funcionamento do algoritmo ICP com e sem a aplicação do método *SetTooFarThreshold*, no alinhamento dos *views* 1 e 15 dos *scans* 1 e 2. Sem a utilização do método *SetTooFarThreshold*, a tranformação *TF1* resultante, posiciona as nuvens de forma a haver a mínima distância possível entre todos os pontos que as constituem. Utilizando o método *SetTooFarThreshold*, a tranformação *TF1* vai posicionar as nuvens de modo a haver a maior coincidência somente entre os pontos da região de sobreposição das nuvens.

Em cada execução do *loop*, o algoritmo ICP vai estimar a transformação *TF1* para alinhar a nuvem *Source* à nuvem *Target*. A transformação *TF1* resultante é então aplicada à nuvem de pontos *Source*, recorrendo ao filtro *Rigid Transform* (ver

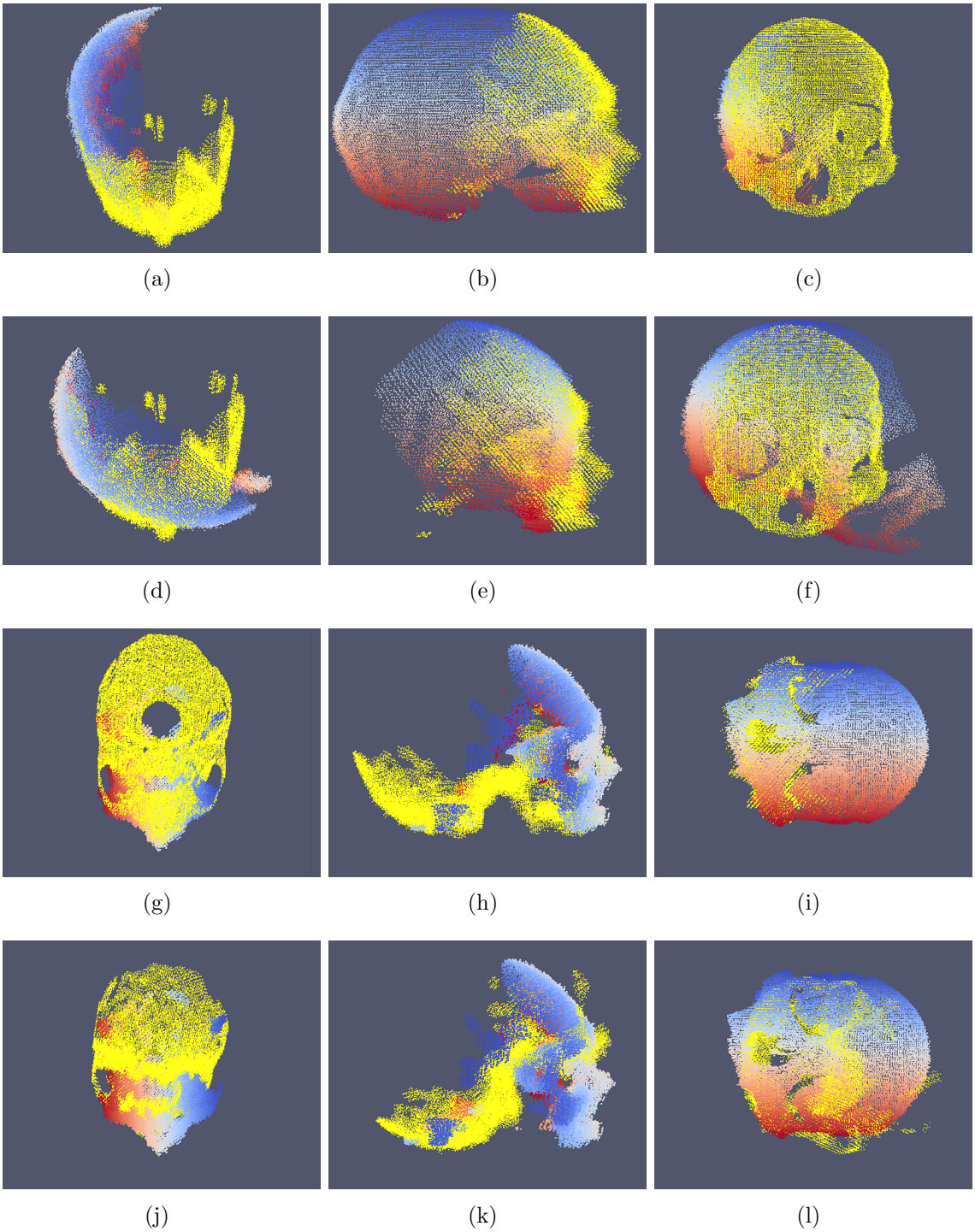


Fig. 6.5: Transformação estimada pelo algoritmo ICP, com e sem o método *SetTooFarThreshold*. (a), (b) e (c) *Scan* 1, com *SetTooFarThreshold* e (b), (d) e (f) sem *SetTooFarThreshold*. (g), (h) e (i) *Scan* 2, com *SetTooFarThreshold* e (j), (k) e (l) sem *SetTooFarThreshold*. Em todos os casos β=100, ρ=2000 e η=20 mm.

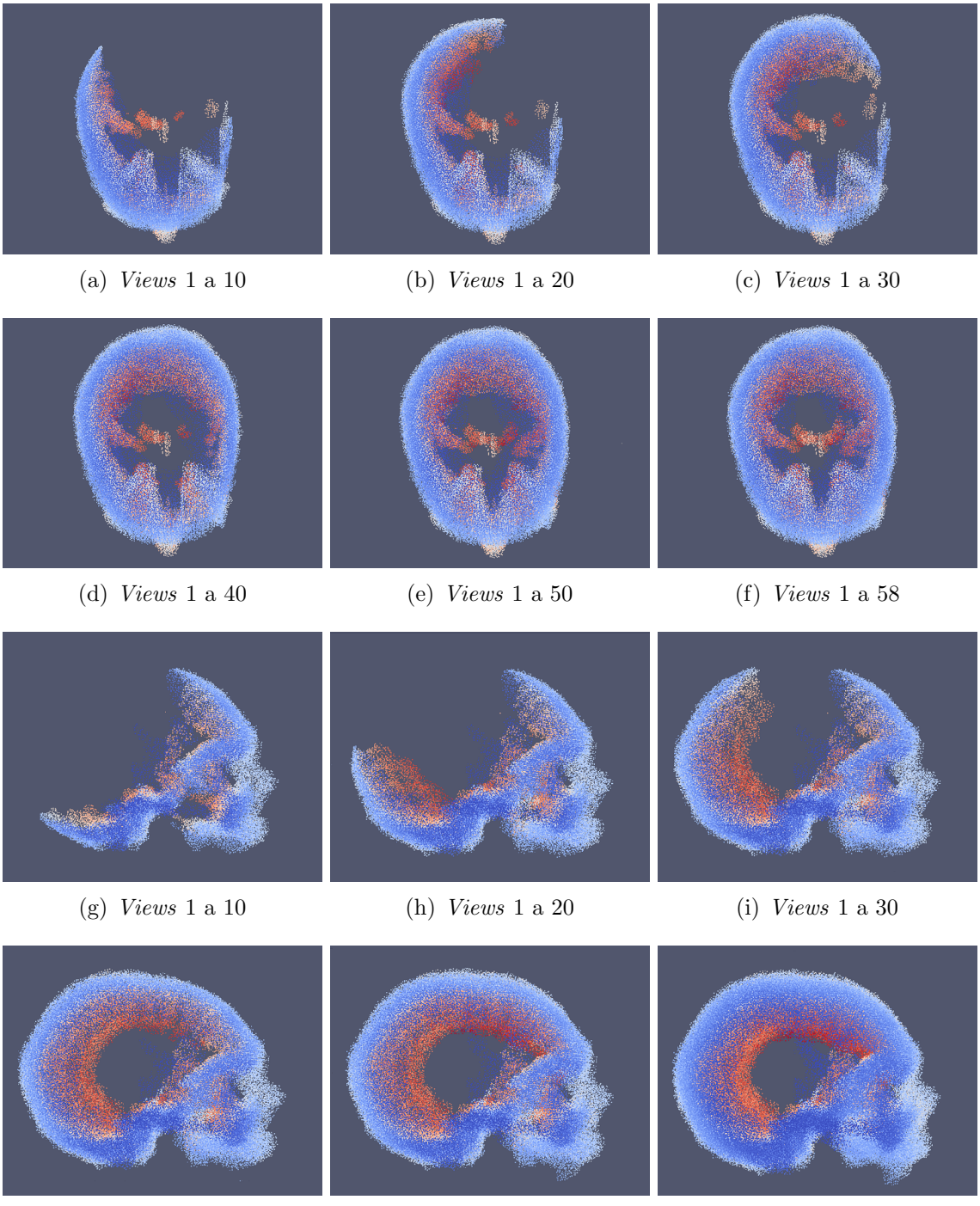
secção 6.2.1). Da mesma forma, a transformação *TF1* também é aplicada à nuvem de pontos não sub-amostrada, como se pode verificar na figura 6.2.

Para se originar a nuvem de pontos global, com informação de toda a superfície do objecto em estudo, é necessário integrar os pontos de cada nuvem alinhada. Para tal, foi implementado um filtro designado *Append*, recorrendo à classe vtkAppendPolyData. Este filtro aceita duas nuvens de pontos através do método *AddInput*, e junta os seus pontos originando uma nuvem maior à saída. Em cada execução do *loop*, os pontos da nuvem *Source* já alinhada, são adicionados aos pontos da nuvem *Target*, pelo filtro *Append*. Para evitar que este conjunto de pontos fique demasiado denso e sejam necessários excessivos recursos computacionais para o processar, é aplicado o filtro de sub-amostragem *Downsample* (ver secção 5.6.2). A saída deste filtro é usada para re-alimentar a nuvem de pontos *Target*, que assim vai adquirindo informação de toda a região do objecto de estudo. Como consequência, em cada execução do *loop*, cada nuvem *Source* é alinhada com o conjunto de todos os pontos já alinhados até à altura. A figura 6.6 apresenta o crescimento da nuvem de pontos *Target*, nos *scans* 1 e 2, ao longo de diferentes execuções do *loop*, com intervalo de leitura (*Gap*) igual a 3.

6.2.3 *Splatting*

Uma das saídas do algoritmo AI é uma imagem 3D com a estrutura do objecto em estudo. Para tal, é definida uma imagem 3D e é implementado um filtro designado *Splatting*. Este filtro introduz na imagem, a informação contida nas nuvens não sub-amostradas, através do mapeamento entre as coordenadas físicas dos pontos nas nuvens e os índices dos pixels na imagem. A imagem 3D é definida através do método descrito no *ITK Software Guide* [52], indicando o seu tamanho, ou seja, o número de pixels segundo cada direcção (N_X, N_Y, N_Z), o espaçamento entre pixels (d_X, d_Y, d_Z), em mm, e a sua posição relativamente ao sistema de coordenadas da câmara (O), em mm. Multiplicando o número de pixels pelo espaçamento entre eles, obtêm-se as dimensões físicas da imagem $(\Delta_X, \Delta_Y, \Delta_Z)$, em mm. Na figura 6.7 pode-se verificar os principais conceitos geométricos associados à imagem ITK.

A posição de um pixel na imagem é identificada por um ID único segundo cada direcção da imagem. O espaçamento entre pixels é medido a partir dos seus centros, representados como círculos, e pode ser diferente segundo cada dimensão da imagem. A origem da imagem está associada às coordenadas do primeiro pixel da imagem. Um pixel é considerado como a região rectangular em torno do centro do



(a) *Views* 1 a 10 (b) *Views* 1 a 20 (c) *Views* 1 a 30

(d) *Views* 1 a 40 (e) *Views* 1 a 50 (f) *Views* 1 a 58

(g) *Views* 1 a 10 (h) *Views* 1 a 20 (i) *Views* 1 a 30

(j) *Views* 1 a 40 (k) *Views* 1 a 50 (l) *Views* 1 a 58

Fig. 6.6: *Append* de nuvens de pontos em diferentes execuções do *loop*. (a) a (f), *Scan* 1, (g) a (l) *Scan* 2.

pixel, que detém determinado valor. Para se definir a imagem 3D considerou-se a transformação *TF0* (descrita na secção 6.2.1) que, em cada nuvem de pontos, colo-

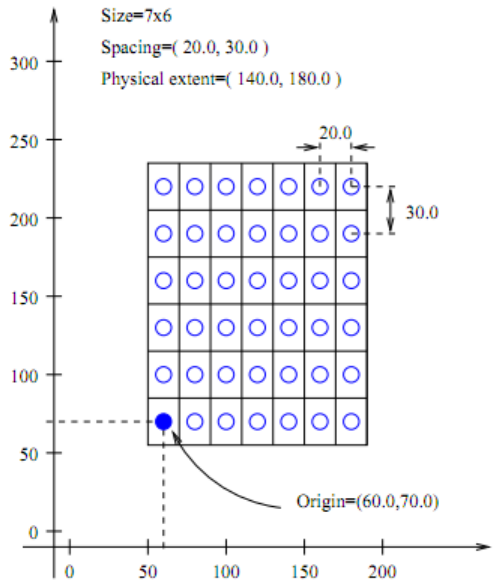


Fig. 6.7: Principais conceitos geométricos da imagem *ITK*.

cou a região do centro da plataforma rotativa na origem do sistema de coordenadas da câmara. Teve-se também em linha de conta que a plataforma rotativa tem 250 mm de comprimento x 250 mm de largura. Assim, definindo uma imagem com dimensões físicas suficientes para abranger a área delimitada pela superfície da plataforma, e centrando-a na origem do sistema de coordenadas da câmara, é possível fazer o mapeamento entre os pontos do objecto em estudo e os pixels da imagem 3D. Definiu-se assim uma imagem 3D com as seguintes características:

- $(N_X, N_Y, N_Z) = (128, 128, 128)$(pixels);
- $(d_X, d_Y, d_Z) = (2, 2, 2)$(mm);
- $(\Delta_X, \Delta_Y, \Delta_Z)= (256, 256, 256)$(mm);
- $O(x, y, z) =$ (-128, -80, -128).

Com a imagem 3D definida, o mapeamento entre pontos das nuvens e os pixels da imagem foi conseguido através do filtro *Splatting*, usando o método *TransformPhysicalPointToIndex* da classe itk::image. Para cada ponto numa nuvem, este método procura o índice do pixel mais próximo na imagem, retornando um *booleano* que indica se o ponto é ou não atribuído a um pixel na imagem. Os pixels na imagem assumem valores escalares, e se um ponto estiver compreendido na região definida pela imagem, o valor do pixel ao qual é atribuído aumenta uma unidade. Assim, por

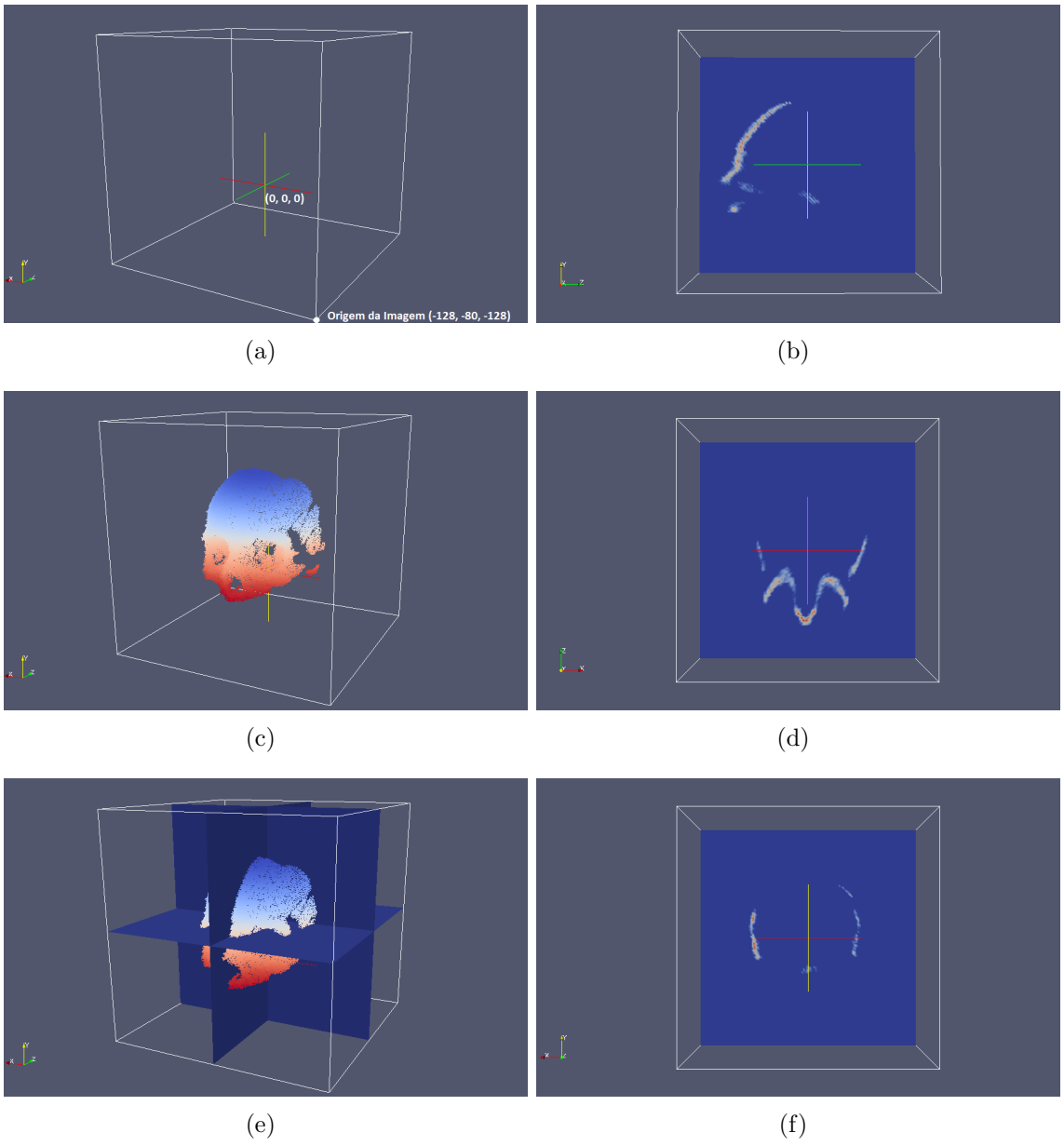
exemplo, um pixel da imagem ao qual sejam atribuídos cinco pontos provenientes das nuvens, vai assumir o valor 5.

O filtro *Splatting* é aplicado a cada nuvem lida pelo algoritmo AI, orientada e alinhada, mas não sub-amostrada. Deste modo, a imagem 3D recebe todos os pontos das nuvens, ao contrário da nuvem de pontos global, que só recebe pontos das nuvens sub-amostradas. No final, a imagem terá recebido pontos de cada uma das nuvens lidas, sendo que os pixels aos quais tiverem sido associados mais pontos, terão um valor mais alto, e os pixels aos quais tiverem sido associados menos pontos, terão um valor mais baixo. Nas figuras 6.8 e 6.9 é possível verificar o posicionamento da imagem 3D em relação à origem do sistema de coordenadas da câmara, e em relação a uma nuvem de pontos do crânio (*view* 1, dos *scans* 1 e 2). Em cada uma das figuras, a imagem definida contém pontos do *view* 1.

Na figura 6.10 é apresentado um *zoom* à informação na imagem 3D gerada com pontos do *scan* 1 e do *scan* 2, ao longo de diferentes execuções do *loop*.

Sendo verificado o critério de paragem do algoritmo AI são produzidos os resultados finais, ou seja, a nuvem de pontos global e a imagem 3D. A nuvem de pontos global é escrita recorrendo à classe vtkpolyDataWriter. Na figura 6.11 são apresentadas as nuvens de pontos globais sob diferentes orientações, obtidas para os *scans* 1 e 2 com *views* 1 a 58, e *Gap*=3. A nuvem global obtida para o *scan* 1 contém 93332 pontos, e a nuvem obtida para o *scan* 2 contém 93192 pontos.

Por sua vez, a imagem 3D é escrita através da classe itk::ImageFileWriter. As figuras 6.12 e 6.13 apresentam cortes segundo planos anatómicos nas imagens 3D obtidas para o *scan* 1 e para o *scan* 2. O valor de cada pixel na imagem 3D é um escalar que corresponde ao número de pontos das nuvens que lhe foram atribuídos. A gama de valores dos pixels da imagem 3D está compreendida entre 0 e 89 para o *scan* 1, e entre 0 e 94 para o *scan* 2.

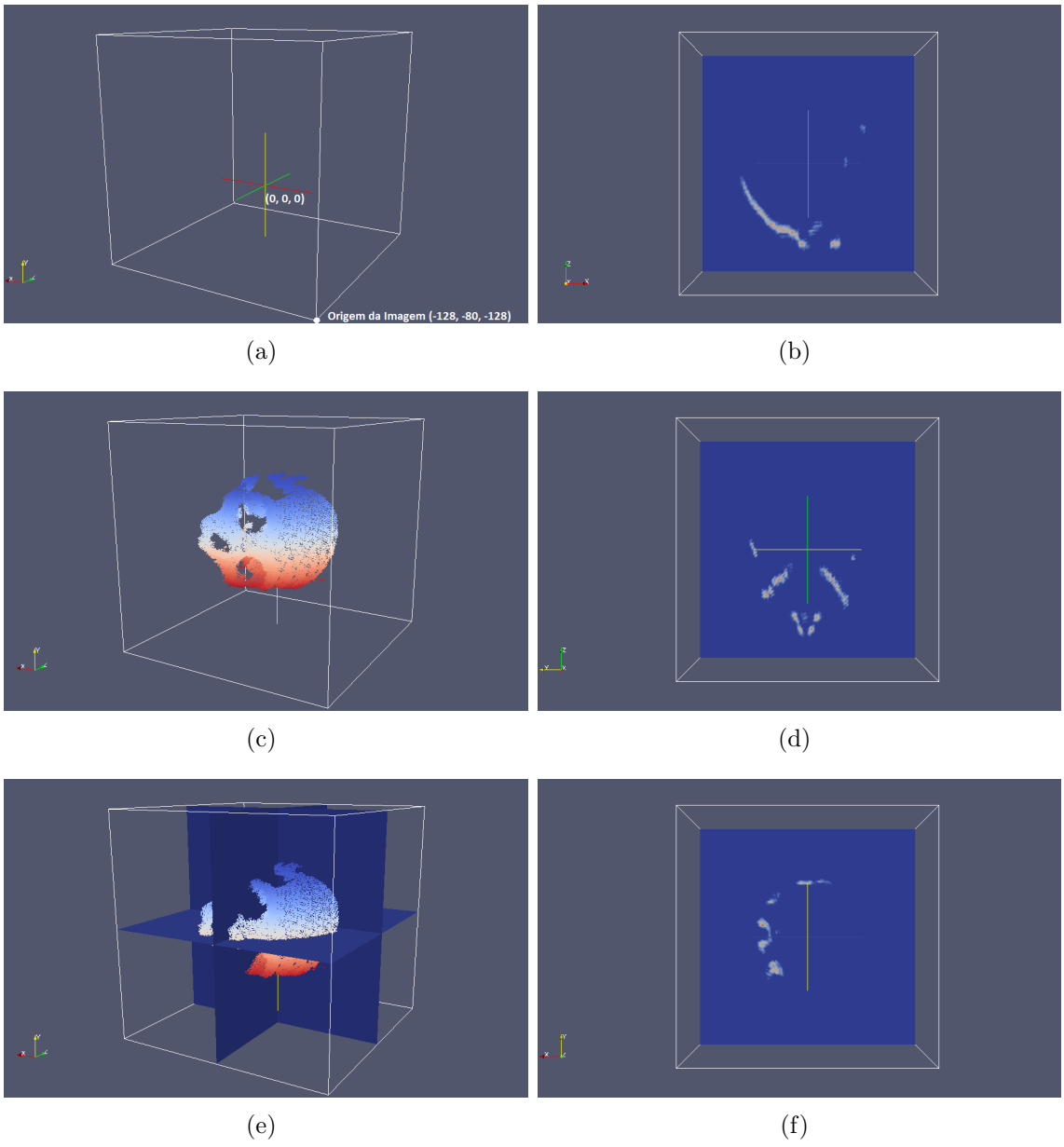


(a) (b)

(c) (d)

(e) (f)

Fig. 6.8: Imagem 3D e *Scan* 1, *view* 1. (a) e (c) Posição da imagem em relação à origem do sistema de coordenadas da câmara e ao *view* 1. (e) Planos na imagem 3D com o *view* 1 sobreposto. (b), (d) e (f) Planos na imagem 3D com pontos do *view* 1, segundo os planos sagital, transversal e coronal.



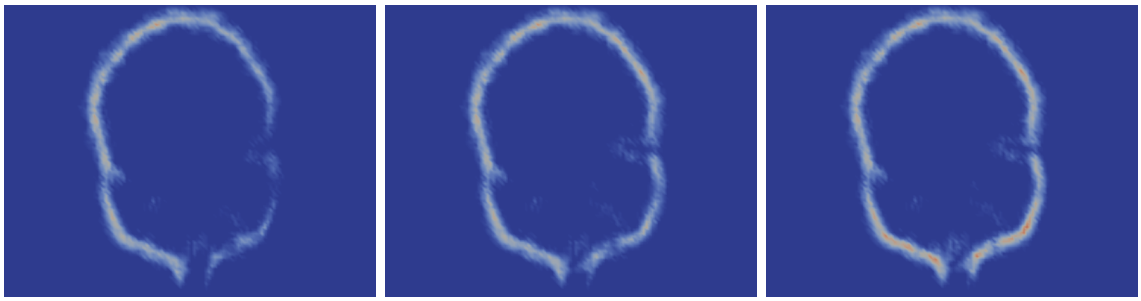
(a) (b)

(c) (d)

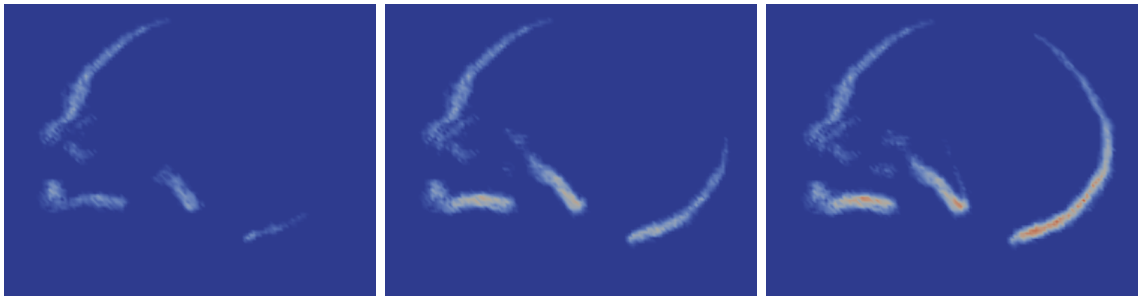
(e) (f)

Fig. 6.9: Imagem 3D e *scan* 2, *view* 1. (a) e (c) Posição da imagem em relação à origem do sistema de coordenadas da câmara e ao *view* 1. (e) Planos na imagem 3D com o *view* 1 sobreposto. (b), (d) e (f) Planos na imagem 3D com pontos do *view* 1, segundo os planos sagital, transversal e coronal.

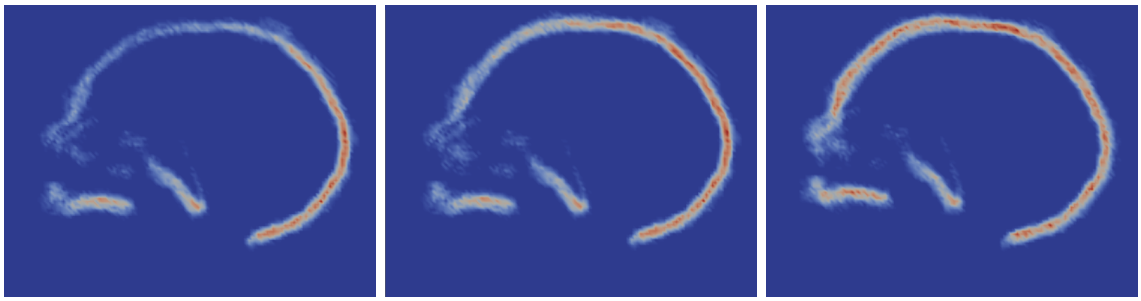
(a) *Views* 1 a 10 (b) *Views* 1 a 20 (c) *Views* 1 a 30



(d) *Views* 1 a 40 (e) *Views* 1 a 50 (f) *Views* 1 a 58



(g) *Views* 1 a 10 (h) *Views* 1 a 20 (i) *Views* 1 a 30



(j) *Views* 1 a 40 (k) *Views* 1 a 50 (l) *Views* 1 a 58

Fig. 6.10: *zoom* à imagem 3D ao longo de diferentes execuções do *loop*, segundo o plano transversal. (a) a (f) *Scan* 1, (g) a (l) *Scan* 2.

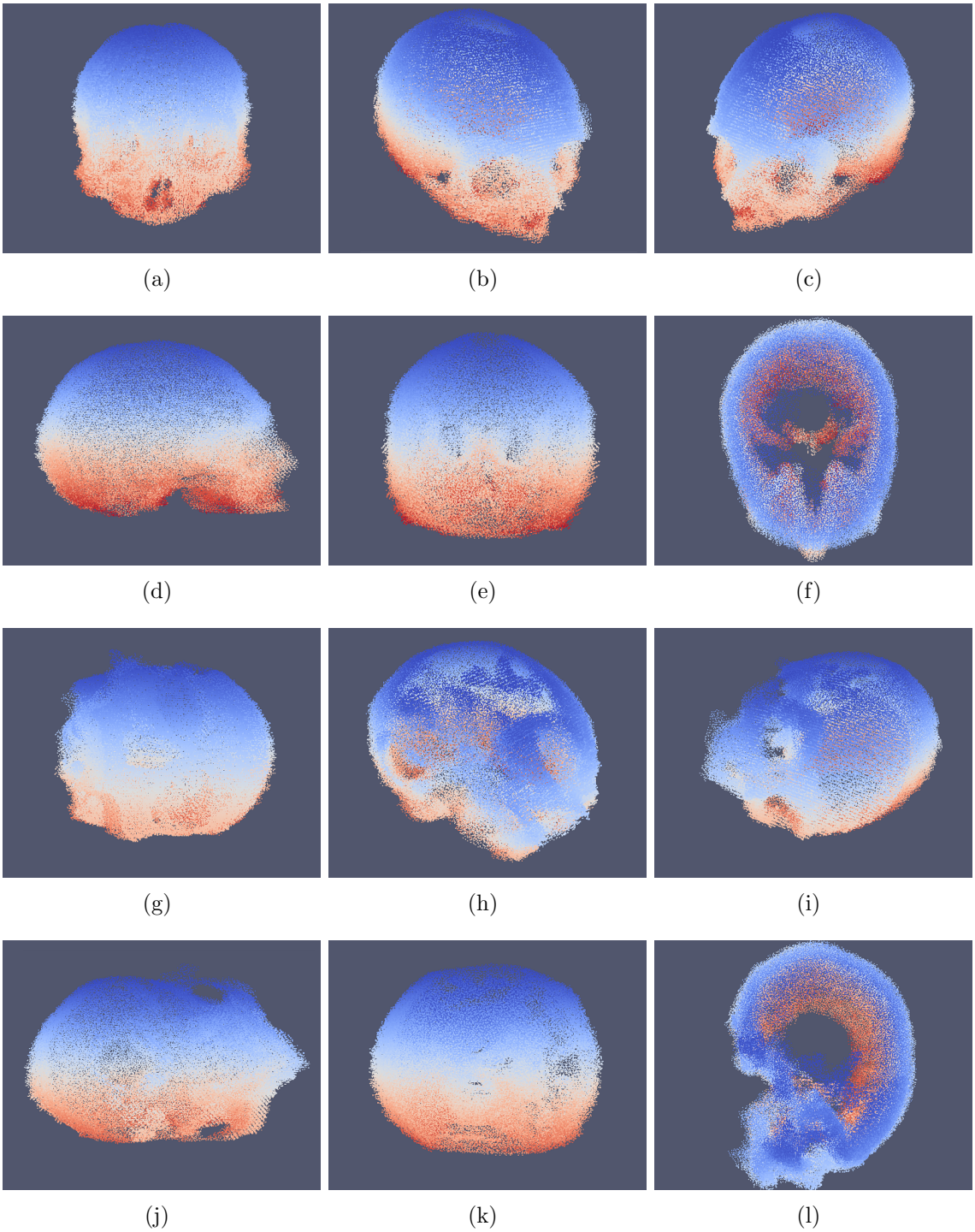
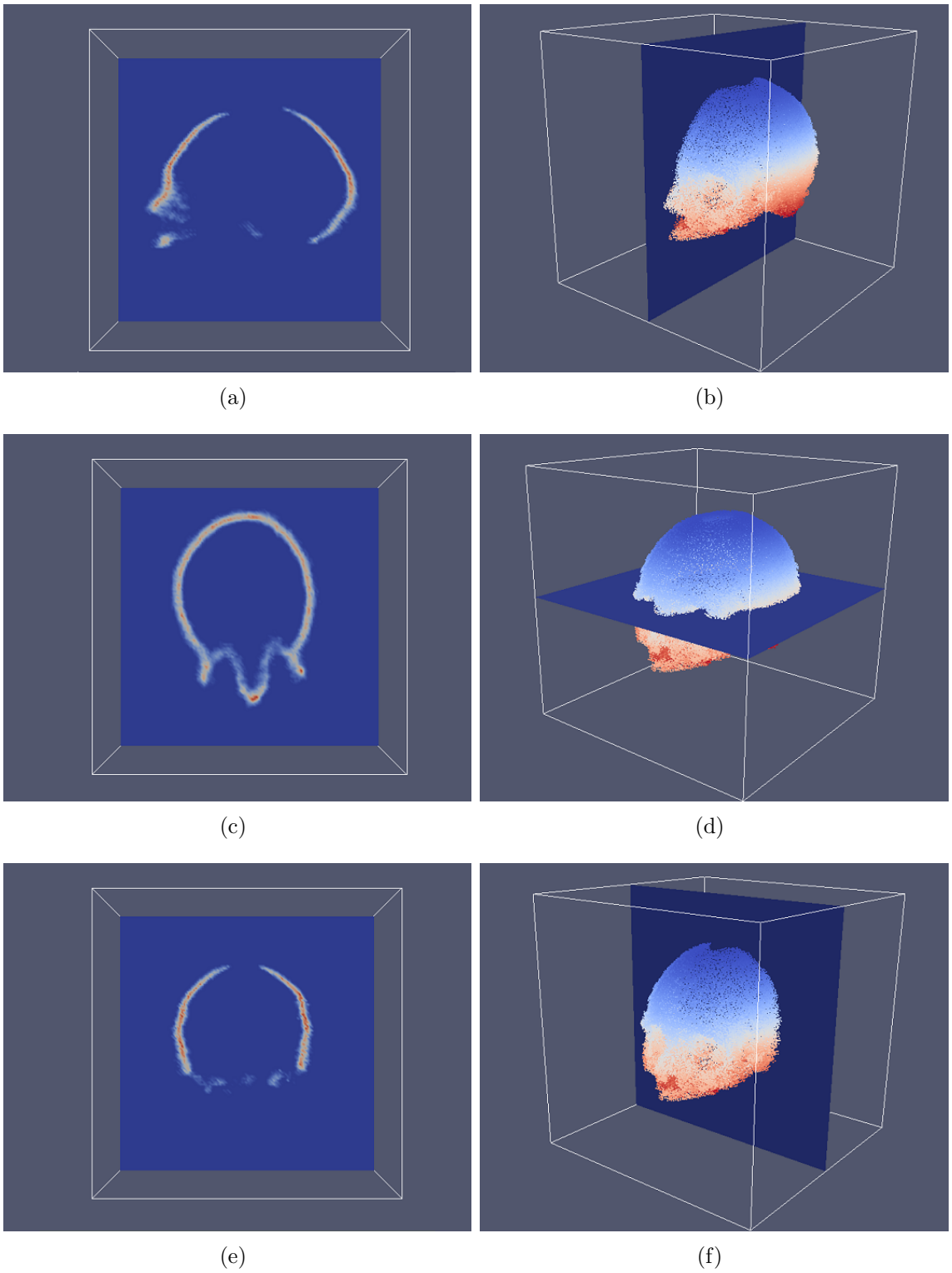


Fig. 6.11: Nuvem de pontos global, com pontos dos *views* 1 a 58 com *Gap*=3, sob diferentes orientações. (a) a (f) *Scan* 1, (g) a (l) *Scan* 2.

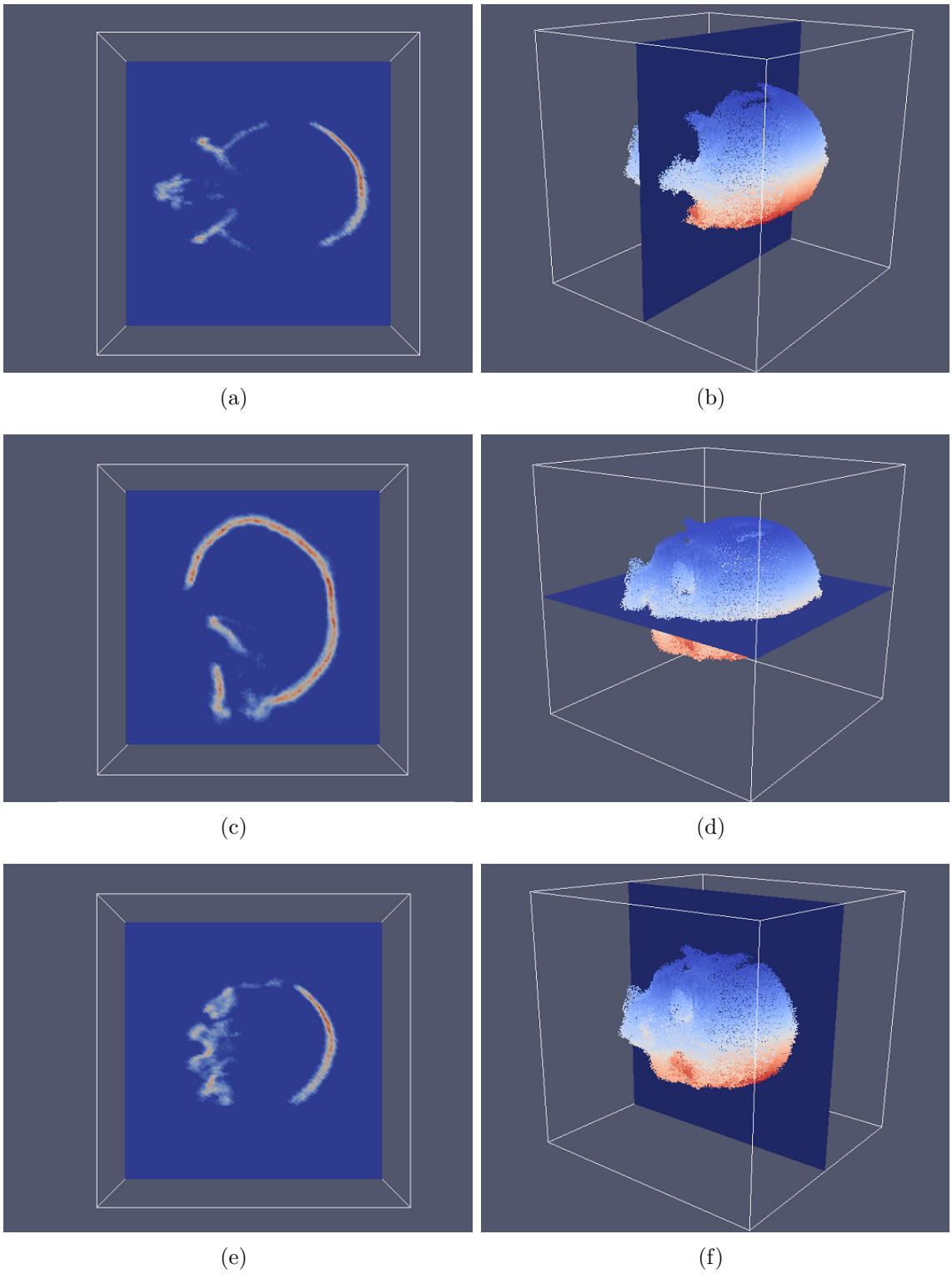


(a) (b)

(c) (d)

(e) (f)

Fig. 6.12: Imagem 3D obtida para o *scan* 1. À esquerda: Cortes na imagem 3D segundo os planos anatómicos. À direita: Correspondência dos cortes com a nuvem de pontos global.



(a) (b) (c) (d) (e) (f)

Fig. 6.13: Imagem 3D obtida para o *scan* 2. À esquerda: Cortes na imagem 3D segundo os planos anatómicos. À direita: Correspondência dos cortes com a nuvem de pontos global.

6.3 Reconstrução de Superfícies

Com o algoritmo AI, obteve-se uma imagem 3D com a estrutura da superfície do crânio analisado pelo *scanner* 3D. O passo seguinte consistiu em utilizar a informação na imagem para gerar uma superfície. Para tal, foi desenvolvido um algoritmo para reconstrução de superfícies, usando VTK. O algoritmo começa por suavizar a imagem através de uma convolução com *kernel* Gaussiano e gera uma superfície recorrendo ao algoritmo *Marching Cubes*. No final produz uma superfície poligonal no formato vtkpolydata, como se pode verificar na figura 6.14.

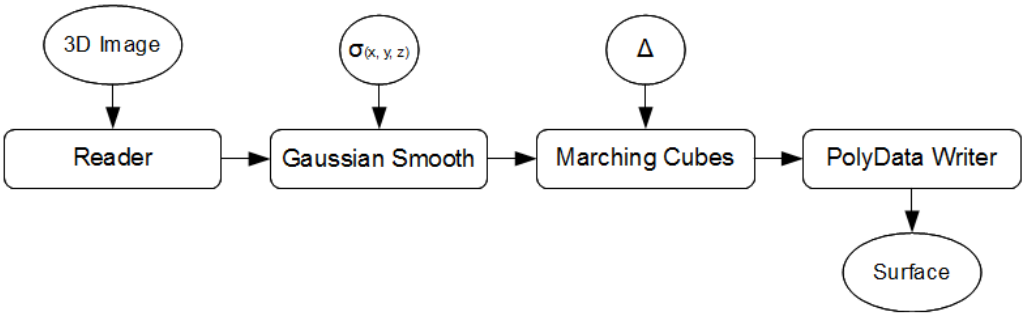


Fig. 6.14: *Pipeline* do algoritmo de reconstrução de superfícies.

Gaussian Smooth

De acordo com a *pipeline* da figura 6.14, a imagem é suavizada através de uma convolução Gaussiana. O objectivo da suavização é diminuir o ruído na imagem de forma a haver uma representação mais coerente da informação nela contida. O filtro *Gaussian Smooth* foi implementado recorrendo à classe *vtkImageGaussianSmooth* e aceita como parâmetro de entrada o desvio padrão da Gaussiana (σ) em unidades de pixels, segundo cada dimensão. É necessário algum cuidado na definição do parâmetro σ, para que se possa remover ruído da imagem e torná-la mais uniforme sem a desbater demasiadamente, podendo até remover-se detalhes e comprometer-se a estrutura representada. Valores mais elevados do parâmetro σ tornam a imagem mais desfocada. Foram então realizados alguns testes com o parâmetro σ nas imagens 3D obtidas nos *scans* 1 e 2. Na figuras 6.15 e 6.16 é possível verificar a influência do parâmetro σ nos valores dos pixels das imagens 3D, apresentadas segundo os mesmos cortes das figuras 6.12 e 6.13, respectivamente.

No final dos testes, o parâmetro σ foi fixado no valor 1 em cada direcção.

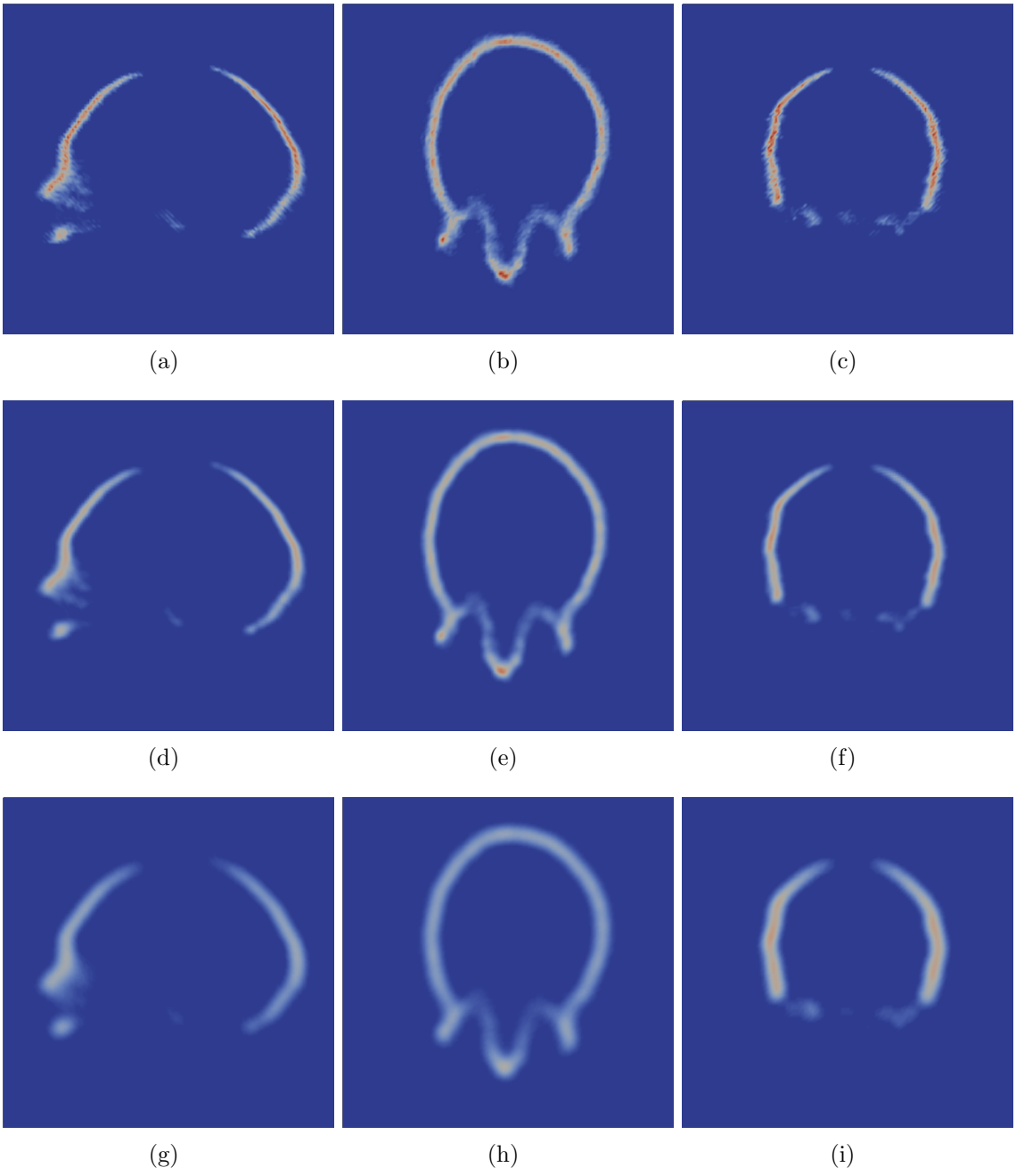


Fig. 6.15: Suavização Gaussiana na imagem 3D do *scan* 1. Suavização Gaussiana na imagem 3D. (a), (b) e (c) Sem suavização, gama de valores: [0, 89]. (d), (e) e (f) Suavização com σ=1, gama de valores: [0, 57]. (g), (h) e (i) Suavização com σ=2, gama de valores: [0, 36].

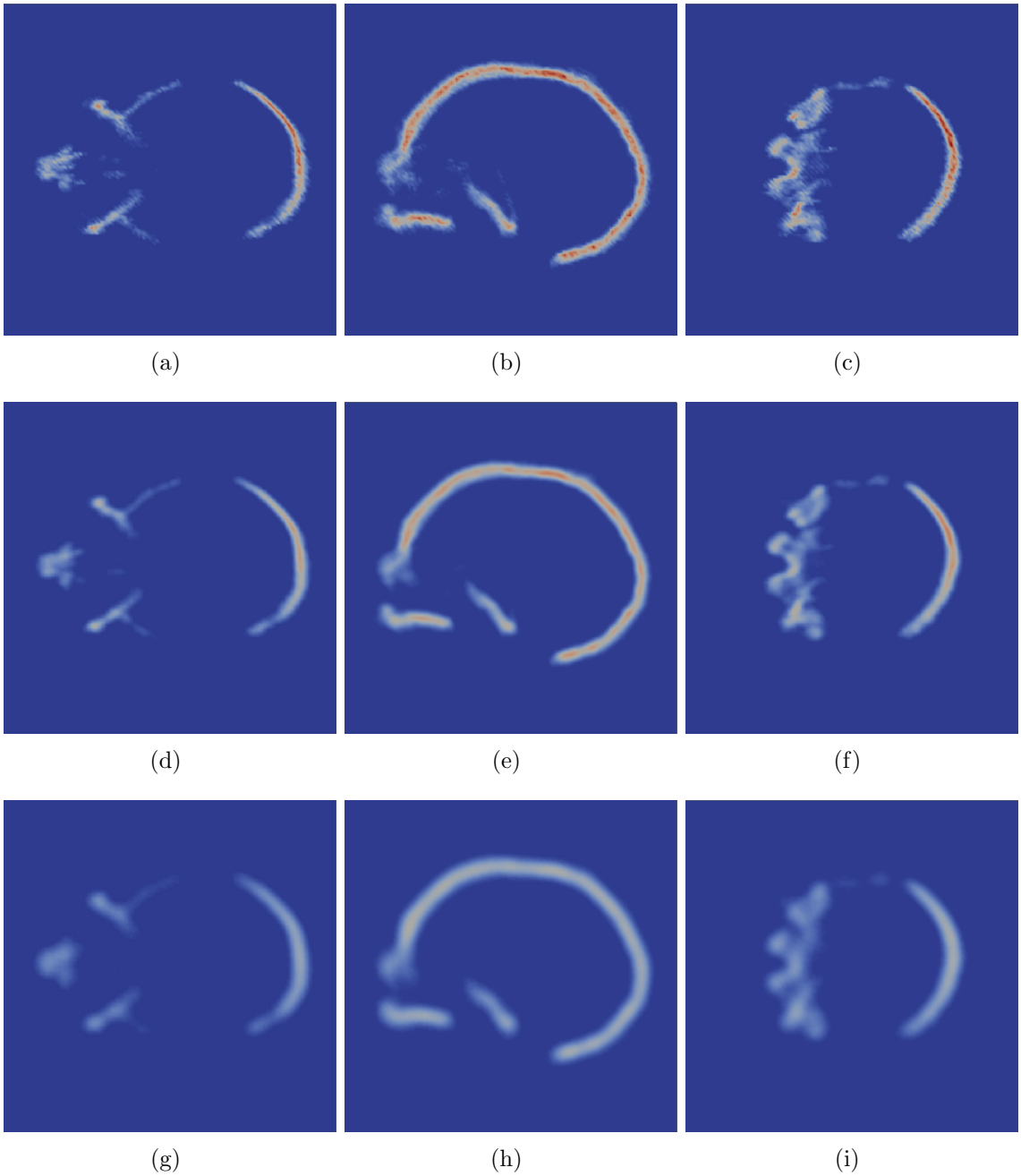


Fig. 6.16: Suavização Gaussiana na imagem 3D do *scan* 2. (a), (b) e (c) Sem suavização, gama de valores: [0, 94]. (d), (e) e (f) Suavização com σ=1, gama de valores: [0, 51]. (g), (h) e (i) Suavização com σ=2, gama de valores: [0, 37].

Marching Cubes

Para se gerar uma superfície a partir da imagem 3D utilizou-se o algoritmo *Marching Cubes*, desenvolvido por William E. Lorensem e Harvey E. Cline, apresentado no SIGGRAPH 1987 [53]. Este algoritmo cria uma representação poligonal de uma superfície, dada pelo conjunto de pixels com o mesmo valor, ou numa gama de valores bem definida (iso-superfície). O algoritmo define um cubo cujos vértices são pixels vizinhos, determina como é que a superfície o intersecta, e "marcha"ao longo de todo o conjunto de pixels da imagem procedendo de igual modo em cada posição (figura 6.17). Para descobrir a intersecção do cubo, é atribuído o valor 1 a cada vértice cujo escalar seja igual ou superior ao valor da superfície que se pretende construir. Estes vértices estão dentro da superfície ou fazem parte dela. Assim, os vértices do cubo com escalares de valor inferior ao da superfície pretendida, recebem o valor 0 e estão fora da superfície. A superfície intersecta as arestas do cubo onde um vértice está dentro (1), da superfície e outro fora (0). Com este critério, é determinada a topologia da superfície dentro de um cubo, sendo possível determinar o polígono necessário para representar a parte da iso-superfície que passa por esse cubo.

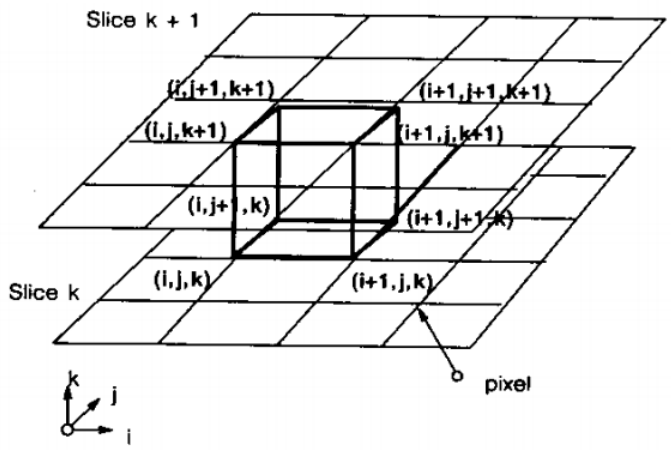


Fig. 6.17: *Maching Cubes.*

Considerando os 8 vértices do cubo, existem $2^8 = 256$ formas da superfície o intersectar. Através de duas simetrias (rotacional e reflexiva), é possível reduzir este número para 15, conforme mostra a figura 6.18. Para cada um dos casos é criado um índice de 8 bits, contendo um bit para o estado (1 ou 0) de cada vértice. Este índice serve como apontador para uma tabela com todas as intersecções possíveis dada a configuração do cubo. Estas intersecções são definidas por polígonos. Por fim, cada vértice dos polígonos gerados é colocado na posição correcta ao longo da aresta do

cubo através da interpolação línear dos dois escalares ligados por essa aresta. Para os algoritmos de visualização definirem as sombras e iluminação da superfície, são também calculadas as normais a cada polígono gerado.

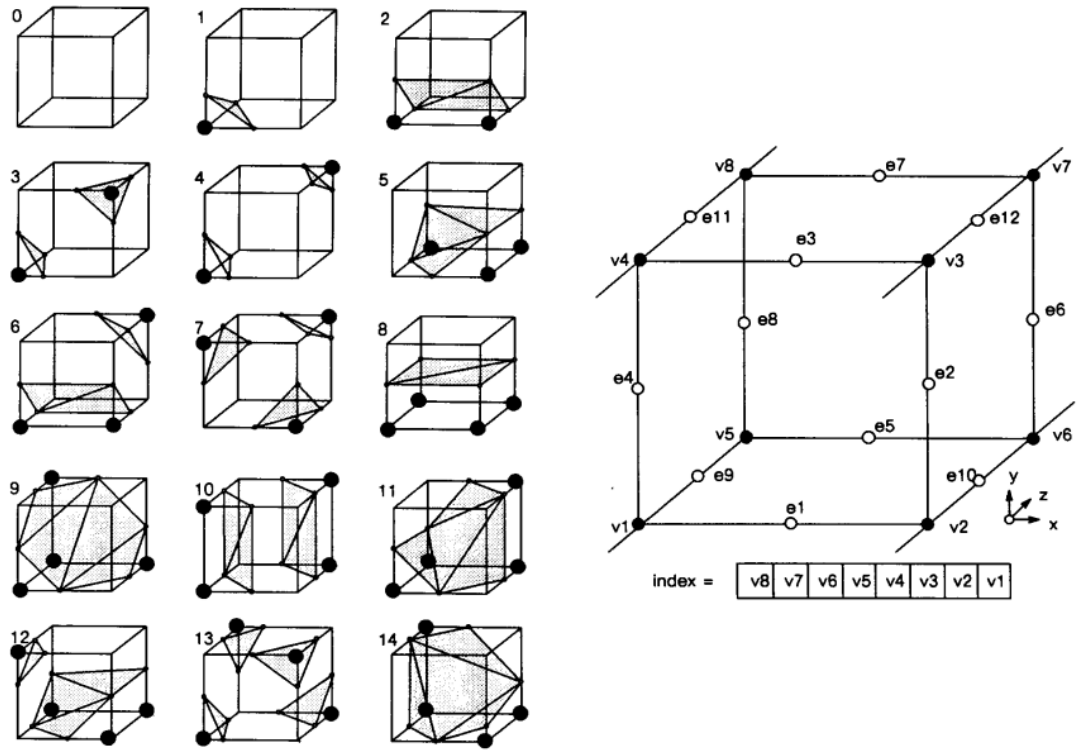


Fig. 6.18: Padrões originais do *Marching Cubes*. Padrões originais de intersecção do cubo e índice para cada caso [53].

No âmbito deste trabalho, o algoritmo *Marching Cubes* foi implementado utilizando a classe *vtkImageMarchingCubes*. Como parâmetro de entrada, é definida uma gama de valores (Δ), nos quais se encontra a superfície pretendida. Por exemplo, se o utilizador definir a gama de valores [5, 10], vai ser gerada uma superfície que passa por todos os pixels da imagem que contenham valores entre 5 e 10. Isto é semelhante a dizer que será criada uma superfície que representará todas as regiões do crânio na imagem onde foram associados entre 5 e 10 pontos das nuvens. Trata-se assim de um problema de optimização, uma vez que o utilizador é que define a superfície a ser reconstruída. No caso de uma imagem de tomografia computorizada, na qual as diferentes estruturas apresentam valores de unidades Hounsfield bem definidas, é possível definir com bastante precisão a iso-superfície que representa a estrutura pretendida. No caso das imagens obtidas neste trabalho assume-se que a superfície real do objecto, neste caso o crânio, passa pelos pixels onde foram introduzidos mais pontos. No entanto, como a densidade de pontos não é idêntica ao longo das ima-

gens, existem pixels que não têm o valor máximo, mas que contêm a superfície real do objecto. Assim, o parâmetro Δ definido para a criação da superfície do objecto tem como limite superior o valor máximo de intensidade dos pixels de cada imagem, variado-se o limite inferior. As figuras 6.19 e 6.20 apresentam alguns testes realizados para os *scans* 1 e 2 para tentar definir uma superfície que represente o crânio analisado. Em cada um dos casos, o tempo necessário para gerar as superfícies correspondeu a 3 segundos.

Procurou-se assim definir o parâmetro Δ de forma a estimar uma superfície o mais aproximada possível à superfície real do crânio. Na figura anterior é possível verificar que o topo da superfície do crânio, na região da bregma e parte da sutura sagital, não se encontra reconstruída. Isto deve-se à posição relativa entre o crânio e a câmara do *scanner* 3D durante as aquisições de pontos, que fez com que a câmara não conseguisse visualizar estas regiões. Como se pode verificar na figura, à medida que o limite inferior do parâmetro Δ aumenta, a superfície tende a ficar mais definida e menor. Isto acontece porque vão sendo considerados cada vez menos pixels, aos quais foram associados pontos mais dispersos. O resultado final deste algoritmo é assim uma superfície estimada sob a influência do parâmetro Δ definido pelo utilizador, obtendo-se superfícies diferentes mediante o parâmetro Δ. A superfície é uma representação poligonal do tipo vtkpolydata. Na tabela 6.1 pode verificar-se o número de células e de pontos em cada uma das superfícies nas figuras 6.19 e 6.20.

Tab. 6.1: Superfícies obtidas com o *Marching Cubes*.

Scan 1			*Scan* 2		
Δ	N.º de Pontos	N.º de Células	Δ	N.º de Pontos	N.º de Células
$[1, 57]$	58982	118004	$[1, 51]$	57944	115940
$[5, 57]$	55122	110296	$[5, 51]$	52664	105372
$[10, 57]$	50950	101940	$[10, 51]$	46108	92248
$[15, 57]$	46344	92676	$[15, 51]$	36890	73768

A figura 6.21 apresenta superfícies obtidas sob dois pontos de vista adicionais, de forma a ter-se uma melhor noção das superfícies desenvolvidas. São também apresentadas as mesmas superfícies mas sem a suavização Gaussiana, de modo a apresentar a sua influência na superfície final.

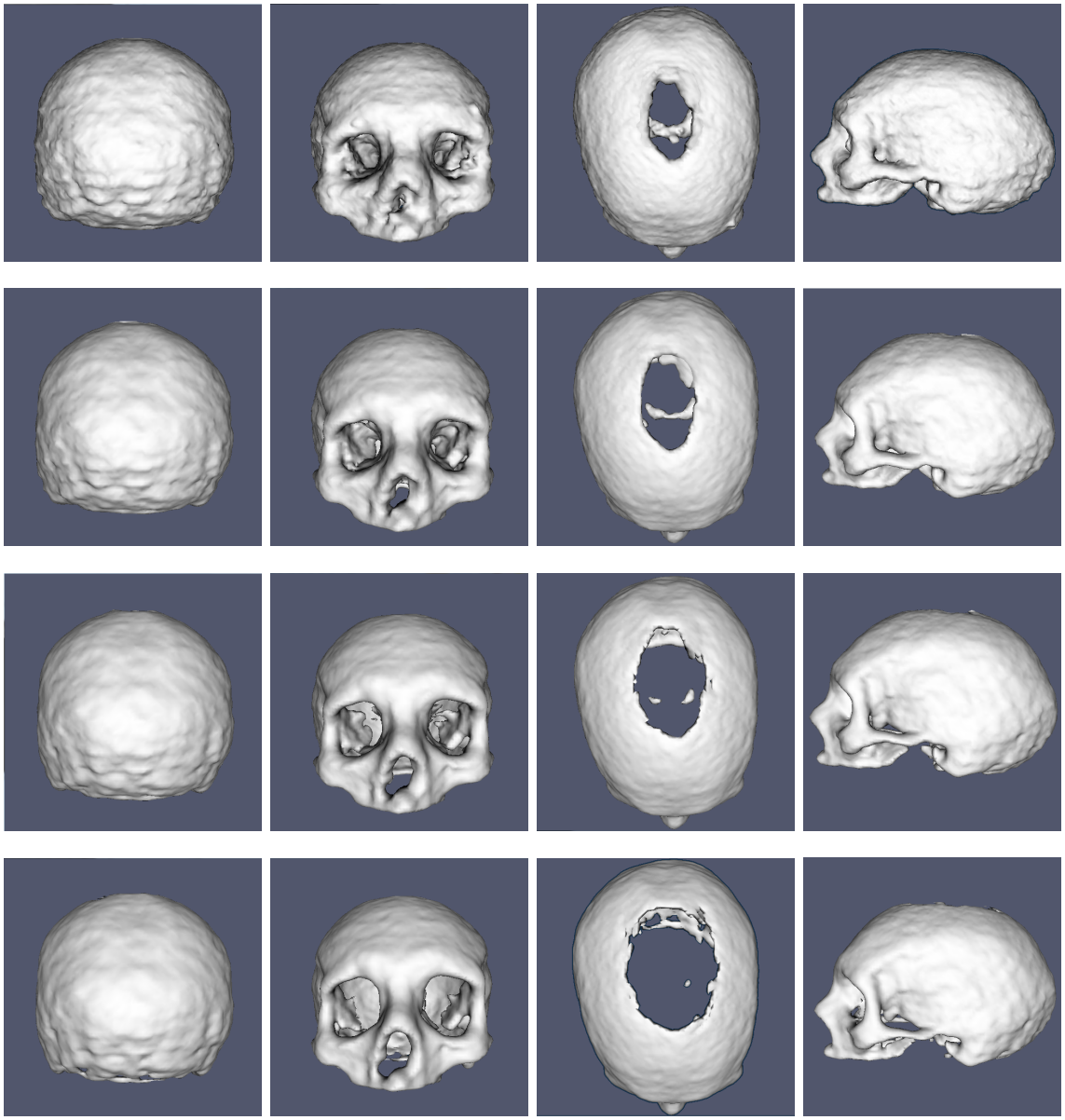


Fig. 6.19: Superfícies geradas pelo *Marching Cubes* para o *scan* 1, com diferentes limites. 1ª linha, Δ=[1, 57]. 2ª linha, Δ=[5, 57]. 3ª linha, Δ=[10, 57]. 4ª linha, Δ=[15, 57] (ver a tabela 6.1).

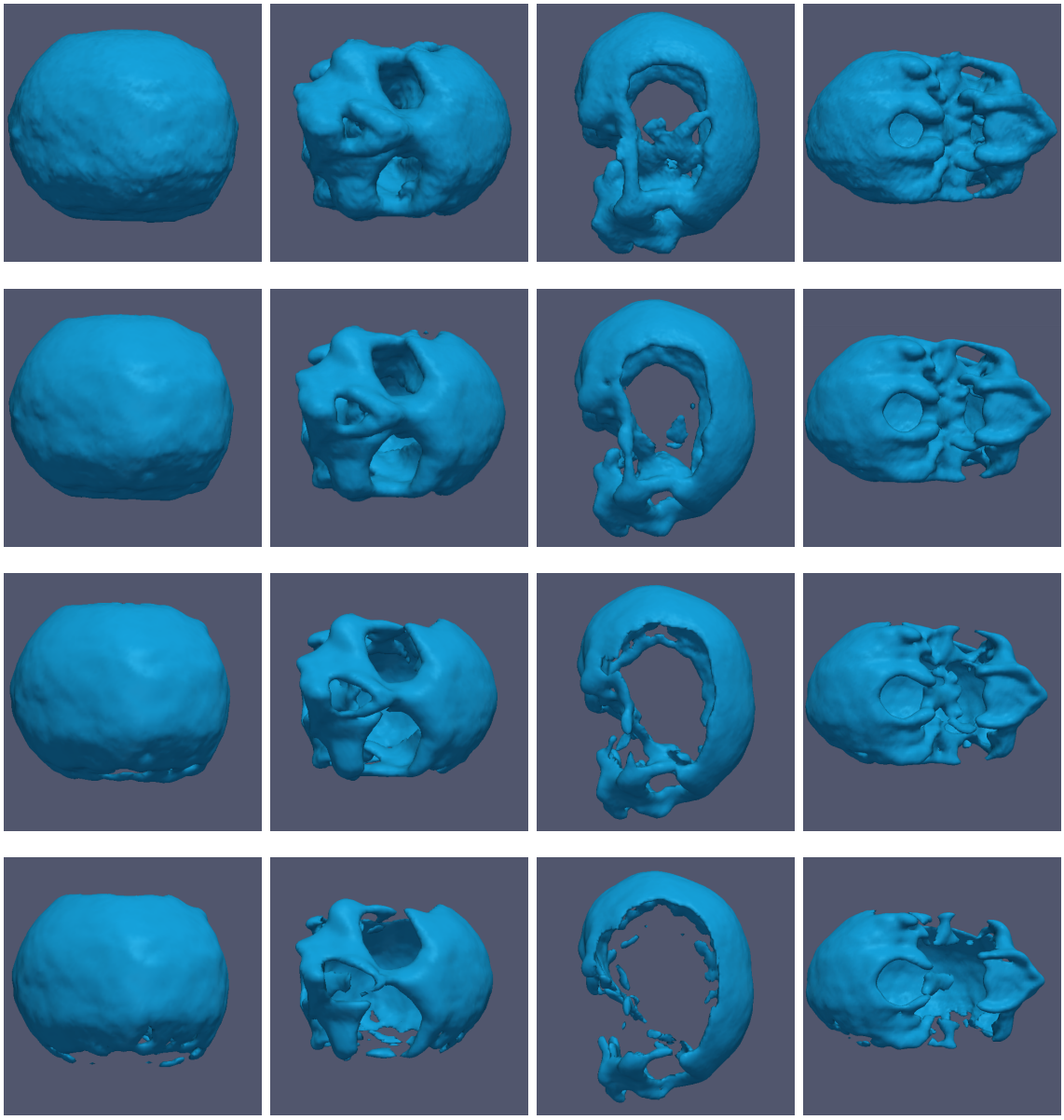


Fig. 6.20: Superfícies geradas pelo *Marching Cubes* para o *scan* 2 com diferentes limites. 1ª linha, Δ=[1, 51]. 2ª linha, Δ=[5, 51]. 3ª linha, Δ=[10, 51]. 4ª linha, Δ=[15, 51] (ver a tabela 6.1).

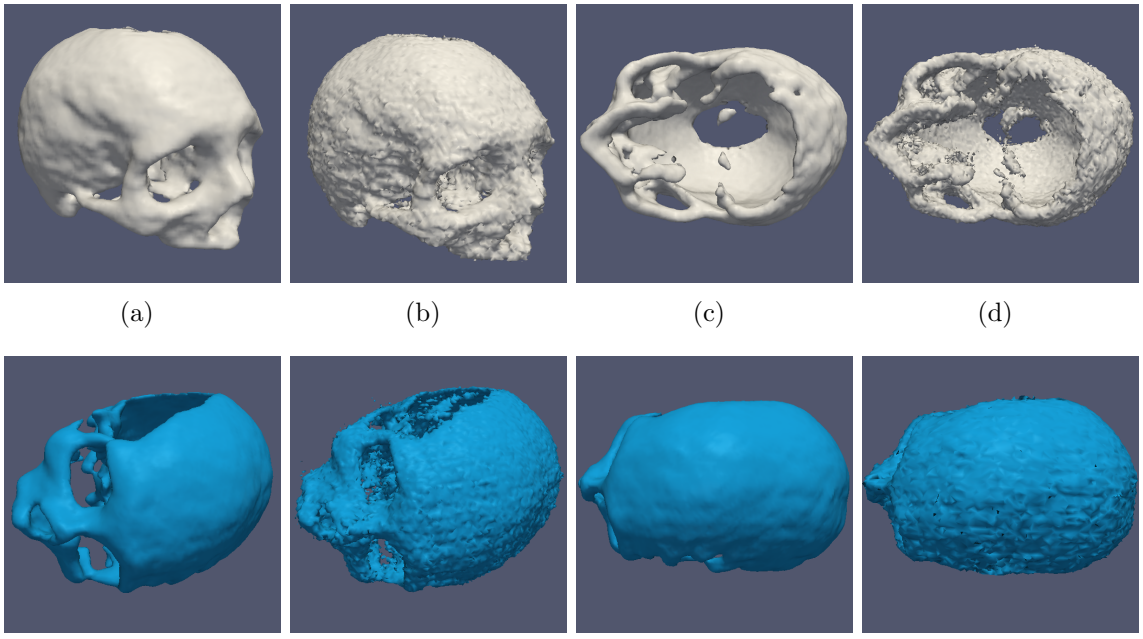
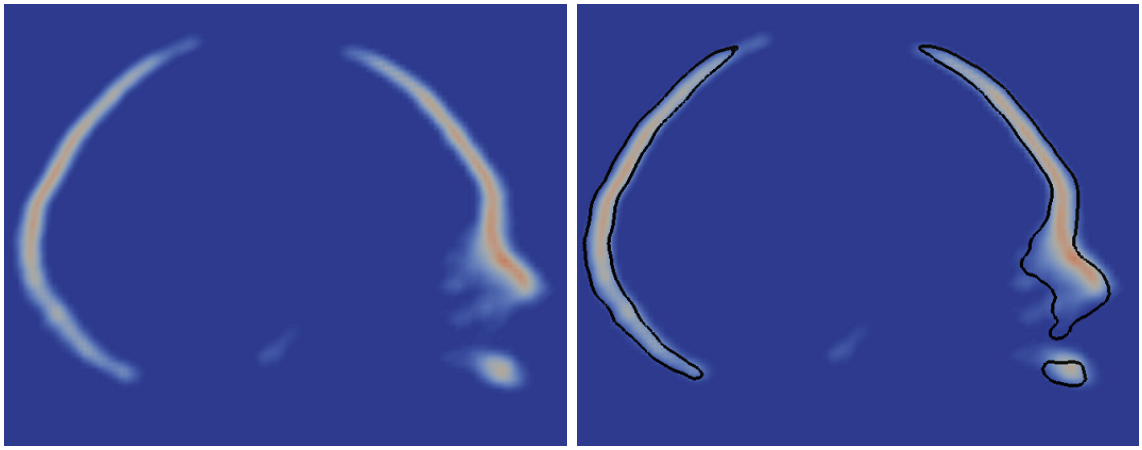


Fig. 6.21: Pormenor de superfícies geradas pelo *Marching Cubes* para os *scans* 1 e 2, com Δ=[10, 57] e Δ=[10, 51], respectivamente. (a) e (c) Com suavização Gaussiana com σ=1. (b) e (d) Sem suavização Gaussiana.

Se as diferentes nuvens de pontos não apresentassem quaisquer pontos dispersos, e se as regiões de sobreposição entre as nuvens coincidissem totalmente, ter-se-ia obtido um conjunto fino de pontos, correspondendo à superfície exacta do crânio. Este conjunto de pontos seria mapeado na imagem, e todos os pixels com valores diferentes de 0 corresponderiam à superfície. Devido à dispersão dos pontos nas nuvens, estes, ao serem mapeados na imagem, fizeram com que os seus pixels apresentassem valores bastante diferentes. Por este motivo, as superfícies originadas apresentam uma parede externa e uma parede interna, como se pode ver na figura 6.22, a título de exemplo. Entre as duas paredes das superfícies encontram-se todas as regiões correspondentes aos pixels da imagem com valor dentro do limite definido por Δ.

A superfície real do objecto encontra-se situada entre estas duas paredes, sendo a superfície obtida uma aproximação à superfície real. Note-se que hoje em dia, a reconstrução 3D de superfícies continua um problema em aberto, sendo constantemente desenvolvidos novos métodos que permitem obter diferentes resultados.

Após a reconstrução das superfícies procedeu-se ao desenvolvimento de um algoritmo para a sua validação. O objectivo deste algoritmo foi permitir quantificar a qualidade das superfícies geradas de modo a ser possível ter uma noção da qualidade



(a) (b)

Fig. 6.22: Paredes interna e externa numa superfície obtida. (a) Corte na imagem 3D. (b) Corte correspondente na superfície e sobreposição à imagem 3D. A superfície real encontra-se situada entre os limites definidos pela superfície originada.

dos resultados obtidos.

6.4 Validação da Reconstrução

Para quantificar a qualidade das superfícies originadas foi desenvolvido um algoritmo de validação. Este algoritmo compara as superfícies do crânio originadas no âmbito deste trabalho com uma superfície originada a partir de uma tomografia computorizada (TC), indicando as distâncias máxima, média e mínima entre regiões correspondentes nas duas superfícies, bem como uma representação visual da distribuição das distâncias ao longo da superfície originada. Desta forma é possível ter uma percepção da qualidade das superfícies originadas em cada região que as constituí, podendo-se ver as regiões melhor e pior representadas. A figura 6.23 apresenta a superfície obtida por TC, utilizada para validação dos resultados.

O algoritmo de validação foi desenvolvido usando VTK e lê duas superfícies do tipo vtkpolydata. Uma vez que as superfícies apresentam um posicionamento diferente, é definida e aplicada uma transformação rígida $TF_{Initial}$, através do filtro *Rigid Transform* (descrito na secção 6.2.1). O objectivo é sobrepôr as superfícies razoavelmente, dando um bom ponto de partida para o algoritmo *Iterative Closest Point* (ver secção 6.2.2). Este algoritmo vai calcular uma segunda transformação TF_{ICP}, para que haja a máxima sobreposição entre pontos correspondentes de ambas as superfícies, alinhando-as o mais precisamente possível. Os parâmetros *Source* e *Target*

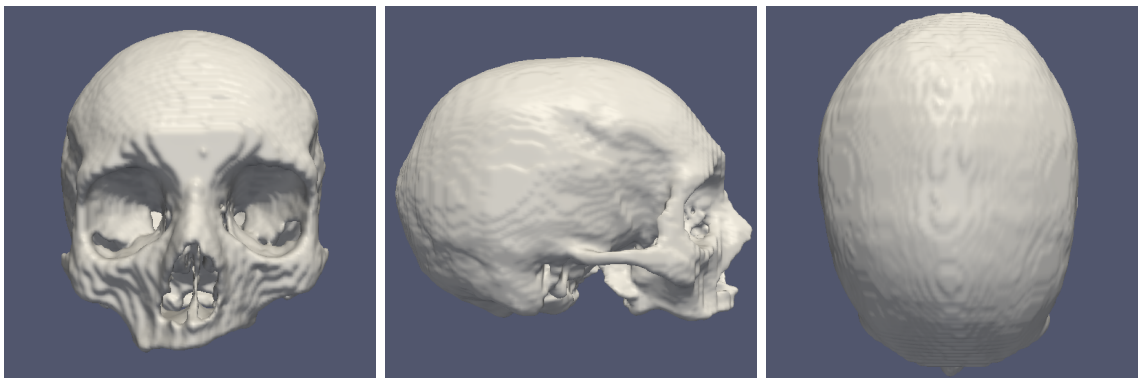


Fig. 6.23: Superfície obtida por TC, usada para validação.

correspondem à superfície originada neste trabalho, e à superfície originada a partir de TC, repectivamente. Após um conjunto de testes, os restantes parâmetros foram definidos com $\beta = 200$, $\rho = 2000$ e $\eta = 30$. A transformação obtida com o *Iterative Closest Point* (TF_{ICP}), é aplicada à superfície *Source*. Com ambas as superfícies alinhadas, procede-se ao cálculo da métrica, ou seja, das distâncias máxima, mínima e média entre os pontos correspondentes nas duas superfícies. Para tal, recorre-se à classe vtkCellLocator e, através do método *FindClosestPoint*, para cada ponto P na superfície *Source*, vai-se procurar pelo ponto Q mais próximo na superfície *Target*, calculando-se a distância $\overline{PQ}$. Em função desta distância, vai ser atribuída uma cor a cada ponto P, da superfície *Source*. O valor da distância entre os pontos correspondentes é introduzido num vector de cor RGB, do tipo vtkUnsignedCharArray e é associado a cada ponto P. No final, é produzida uma superfície do tipo vtkpolydata igual à *Source*, mas colorida, com cada ponto P colorido de acordo com a distância que apresenta ao ponto Q correspondente na superfície *Target*. A figura 6.24 apresenta a *pipeline* do algoritmo de validação.

Este algoritmo foi aplicado às diferentes superfícies originadas neste trabalho, e a métrica obtida para cada uma delas encontra-se na tabela 6.2.

A distância média entre os pontos de cada superfície originada e os pontos da superfície obtida por TC é inferior a 6.5 mm. Relativamente à distâncias máximas, verifica-se uma ligeira variação, sendo que a tendência é que as superfícies com menos pontos apresentem uma distância máxima inferior, à superfície originada por TC. Nas figuras 6.26 e 6.27 é possível ter uma noção visual da distribuição das distâncias nas diferentes regiões de cada superfície originada. Para uma visualização mais fácil, a gama de cores foi reajustada e os resultados são apresentados de acordo com a

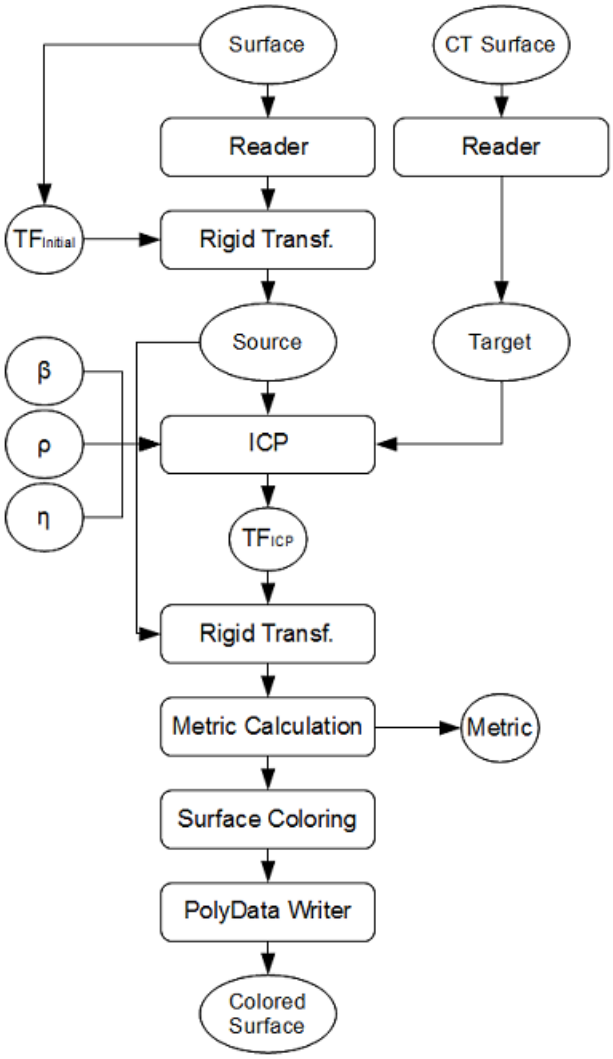


Fig. 6.24: *Pipeline* do algoritmo de validação de superfícies.

escala de cor da figura 6.25.

Tab. 6.2: Métrica (distância) obtida pelo algoritmo *Validation* (em mm).

Scan 1				*Scan* 2			
Δ	D. Máx.	D. Min.	D. Méd.	Δ	D. Máx.	D. Min.	D. Méd.
$[1, 57]$	19.104	0.000	4.548	$[1, 51]$	24.472	0.000	6.143
$[5, 57]$	17.554	0.000	4.180	$[5, 51]$	22.471	0.000	5.763
$[10, 57]$	15.819	0.000	3.896	$[10, 51]$	21.174	0.000	5.594
$[15, 57]$	14.500	0.000	3.742	$[15, 51]$	20.065	0.000	5.716

É possível verificar que todas as superfícies apresentam uma distância maior na região occipital. Na figura 6.28 é possível verificar a sobreposição entre superfícies

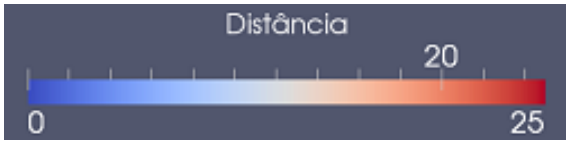


Fig. 6.25: Escala de representação de cor nas superfícies.

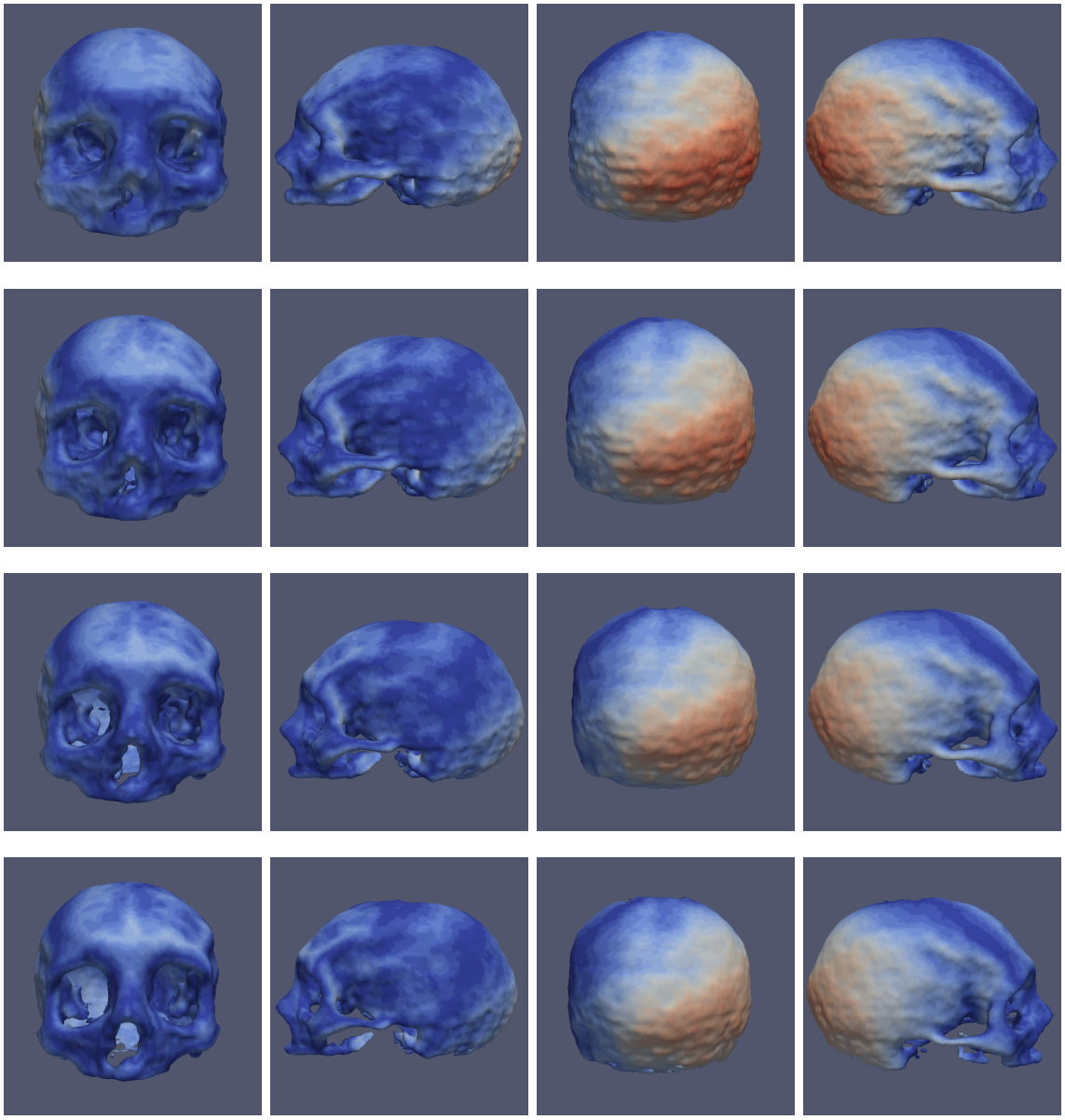


Fig. 6.26: Superfícies obtidas para o *scan* 1, em função da distância à superfície de validação. 1ª linha, Δ=[1, 57]. 2ª linha, Δ=[5, 57]. 3ª linha, Δ=[10, 57]. 4ª linha, Δ=[15, 57]

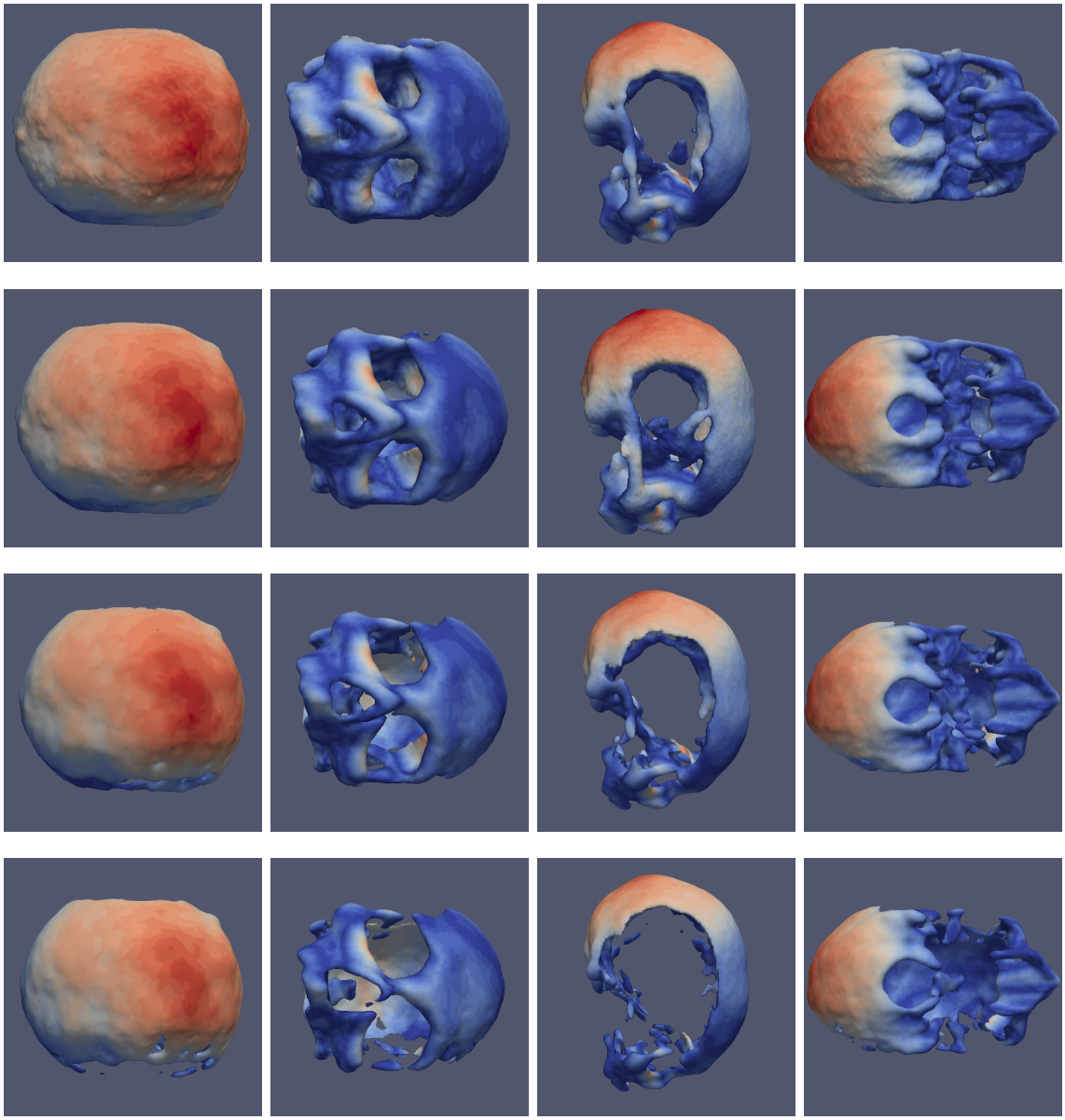


Fig. 6.27: Superfícies obtidas para o *scan* 2, coloridas em função da distância à superfície de validação. 1ª linha, Δ=[1, 51]. 2ª linha, Δ=[5, 51]. 3ª linha, Δ=[10, 51]. 4ª linha, Δ=[15, 51]

obtidas com Δ=[10, 57] para o *scan* 1 e com Δ=[10, 51] para o *scan* 2, e a superfície de validação, de forma a ter uma percepção melhor da diferença entre ambas.

Note-se que as superfícies obtidas neste trabalho apresentam uma parede externa e uma parede interna. Além disso, a superfície de validação também apresenta duas paredes devido à espessura dos ossos do crânio, já que se trata de um exame de TC.

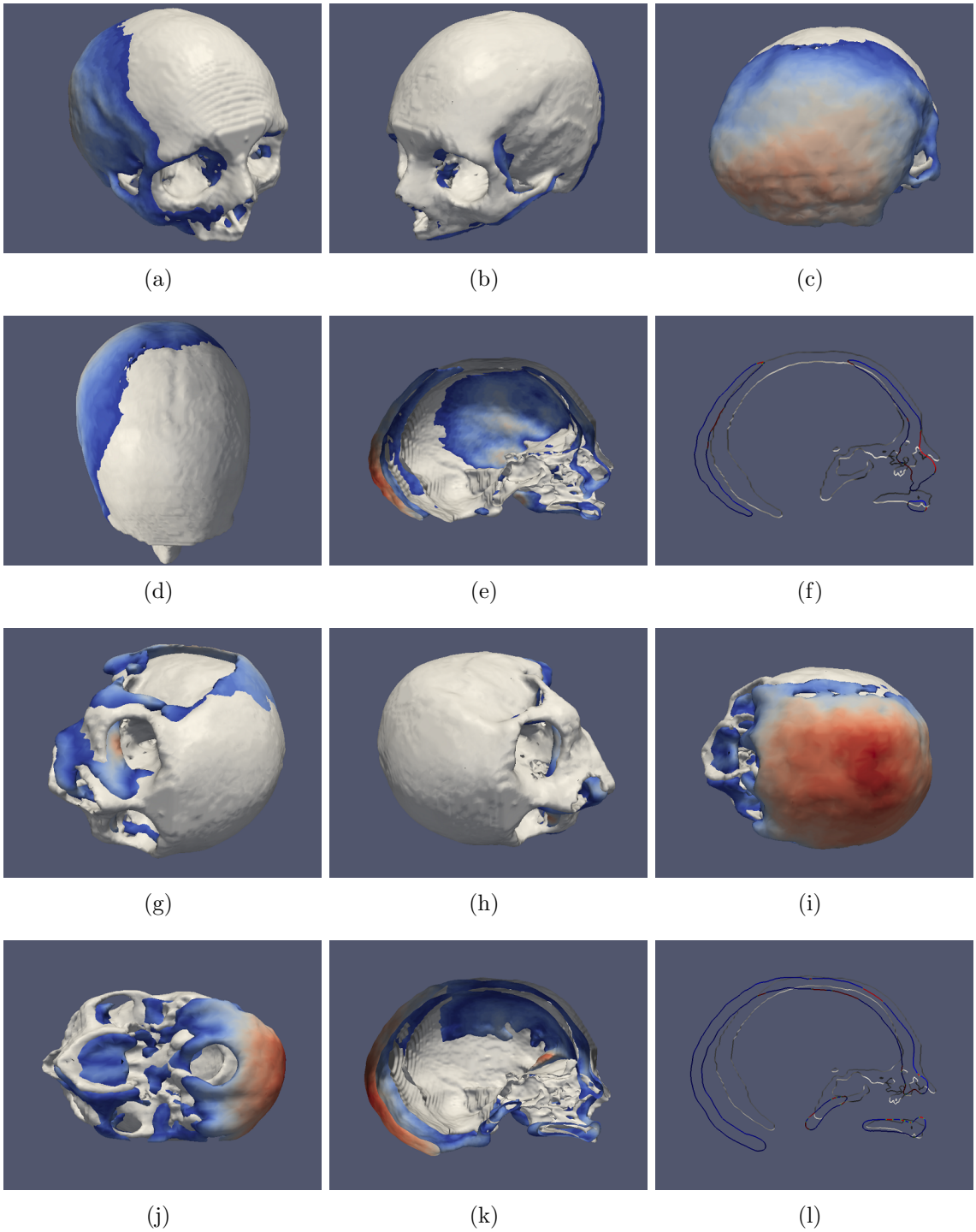


Fig. 6.28: Sobreposição entre superfícies. (a) a (d) Sobreposição entre a superfície obtida para o *scan* 1, com Δ=[10, 57] e a superfície de validação (a branco). (e), (f) Corte e *clip* na superfície do *scan* 1 na superfície de referência. (g) a (j) Sobreposição entre a superfície obtida para o *scan* 2, com Δ=[10, 51] e a superfície de validação (a branco). (k), (l) Corte e *clip* na superfície do *scan* 2 na superfície de referência.

6.5 Resultados e Discussão

Considerando as superfícies obtidas para os dois *scans* realizados, é possível verificar que todas elas apresentam um desfasamento maior na região occipital face à superfície de referência. Uma vez que este desfasamento poderia dever-se ao alinhamento das nuvens de pontos, foram exploradas duas alternativas ao algoritmo AI implementado (secção 6.2). O algoritmo AI alinha as diferentes nuvens de pontos (*views*) por ordem ascendente (*view* 1, *view* 2, *view* 3, etc) e como resultado, o alinhamento e a integração do crânio são realizados da direita para a esquerda (figuras 6.10 e 6.6). A primeira alternativa consistiu no alinhamento dos diferentes *views* por ordem descendente (da esquerda para a direita), e a segunda alternativa consistiu num alinhamento começando pelos *views* mais próximos e seguindo para os *views* mais distantes (ou seja, de frente para trás). No entanto, os resultados obtidos em cada alternativa foram semelhantes aos resultados obtidos com o algoritmo AI apresentado neste trabalho, e as superfícies originadas apresentaram o mesmo desfasamento. É possível afirmar que, ou a região occipital do crânio foi mal adquirida pelo *scanner* 3D em ambos os *scans*, ou a correspondência entre os pontos daquela região específica e os pontos da restante estrutura do crânio não apresentam uma correspondência suficiente para que o algoritmo ICP defina uma transformação aceitável.

Relativamente à reconstrução das superfícies, o algoritmo *Marching Cubes* permitiu estimar rapidamente (em cerca de 3 segundos), superfícies com base num critério Δ, definido pelo utilizador (secção 6.3). A escolha deste parâmetro é limitativa, já que depende do utilizador, e porque faz com que o algoritmo despreze pixels nos quais poderia estar representada parte da superfície. Ou seja, a escolha de dois parâmetros Δ diferentes, resulta em duas superfícies diferentes. As superfícies obtidas também apresentam uma parede externa, e uma parede interna, devido ao espalhamento dos pontos que representam a superfície do objecto. Estes aspectos constituem uma limitação do método adoptado. Foram explorados métodos adicionais, descritos em [45] e [54], para tentar criar uma superfície directamente a partir da nuvem de pontos global. No entanto, a definição de uma topologia somente a partir das coordenadas (x, y, z) dos pontos é demorada e complicada, sendo sensível a regiões de grande curvatura, e à presença de *outliers* e de pontos da superfície afastados uns dos outros [9]. Os métodos testados são agora descritos, sendo que nenhum deles permitiu obter uma superfície que representasse a estrutura do crânio melhor do que a superfície obtida pelo *Marching Cubes*:

- **Tetraedrização de Delaunay**: Este método divide um conjunto de pontos em tetraedros, criando um *convex hull* (o menor conjunto convexo de todos os pontos) a partir dos pontos originais. Este método obedece ao critério de Delaunay para criar tetraedros, sendo que uma esfera que contenha os quatro pontos de um dado tetraedro originado, não pode conter qualquer outro ponto. O tempo necessário para criar os tetraedros a partir da nuvem de pontos global, correspondeu a cerca de 7 minutos. A figura 6.29 apresenta as superfícies obtidas para os *scans* 1 e 2, utilizando a tetraedrização de Delaunay.

- ***Gaussian Splatting***: Este método consiste em injectar os pontos da nuvem de pontos global numa grelha estruturada 3D. Quando um ponto é injectado num pixel da grelha, é espalhado pelos pixels vizinhos. Os pontos são espalhados segundo uma distribuição Gaussiana, originando "esferas". De seguida, é originada uma iso-superfície considerando toda a gama de valores dos pixels na grelha definida. Este método apresenta uma desvantagem, já que os pontos são injectados numa grelha 3D sem relações com unidades físicas e sistemas de coordenadas físicos, ao contrário da imagem 3D. O algoritmo *Gaussian Splatting* demora cerca de 30 segundos a gerar a superfície 3D, e na figura 6.29 é possível ver as superfícies obtidas para os *scans* 1 e 2, utilizando este método.

- ***Surface From Unorganized Points***: O terceiro método explorado consistiu noutro exemplo do VTK, baseado nas classes vtkReconstructionFilter e vtkContourFilter. Este método também considera os pontos numa nuvem e origina as respectivas superfícies. No entanto, não foram atingidos bons resultados nos diferentes testes realizados, e o tempo de processamento foi extremamente longo, correspondendo a cerca de 25 minutos. A figura 6.29 apresenta as superfícies obtida para os *scans* 1 e 2, utilizando este método.

Relativamente ao algoritmo de validação, este constitui um bom método para alinhar e comparar duas superfícies. No entanto, é necessário considerar que a superfície obtida neste trabalho apresenta uma parede externa e uma parede interna. Além disso, a superfície de referência também apresenta duas paredes devido à espessura dos ossos do crânio, já que se trata de um exame de TC. Este factor influencia a transformação estimada pelo algoritmo ICP. Mesmo que só fosse considerada a fronteira exterior da superfície de referência, o algoritmo tenderia a posicionar esta fronteira entre as paredes externa e interna da superfície obtida. Como não se sabe exactamente onde está situada a superfície real do crânio entre as superfícies ex-

terna e interna obtidas, o alinhamento ideal só poderia ser deterimado quando esta questão fosse resolvida.

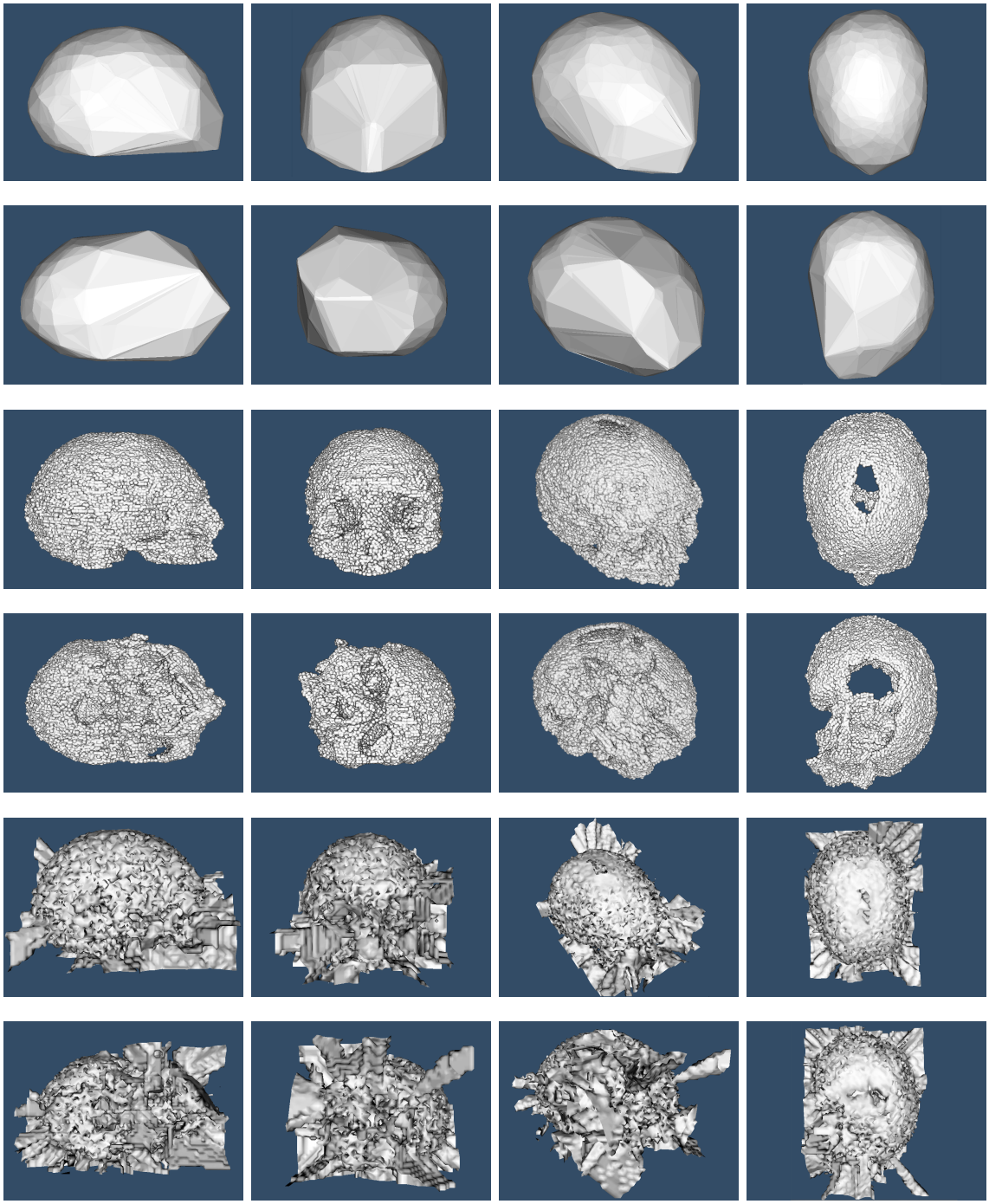


Fig. 6.29: Superfícies obtidas usando métodos alternativos, para os *scans* 1 e 2. Linhas 1 e 2: Tetraedrização de Delaunay, linhas 3 e 4: *Gaussian Splatting*, linhas 5 e 6: *Surface From Unorganized Points*.

Dados os resultados obtidos no âmbito deste trabalho, é possível, no entanto, ter uma percepção visual da qualidade das regiões reconstruídas e ter uma percepção de, dentro das superfícies originadas, qual a que apresenta melhor resultado.

Capítulo 7

Conclusão

A importância e o valor dos *scanners* 3D têm-se evidenciado ao longo dos últimos anos, o que pode ser comprovado pelo aumento do número de sistemas e de técnicas existentes. Esta importância é transversal a várias áreas como a Medicina, a Arqueologia e a Indústria cinematográfica, nas quais as aplicações dos *scanners* 3D têm aumentado e têm contribuído para o avanço destas áreas. Apesar da grande evolução dos *scanners* 3D, a aquisição de pontos 3D e a reconstrução de superfícies constituem ainda hoje um problema em aberto, sendo explorados diferentes métodos para encontrar a melhor solução possível.

Neste trabalho desenvolveu-se um *scanner* 3D e um conjunto de algoritmos que constituem uma abordagem simples e económica à aquisição de informação 3D de objectos e ao problema da reconstrução de superfícies 3D. O *scanner* 3D foi utilizado para adquirir pontos da superfície de um crânio sob diferentes orientações. Graças à plataforma rotativa, o processo de aquisição foi automatizado, evitando o reposicionamento manual do crânio e reduzindo fontes de erro. Adicionalmente, uma vez que a plataforma rotativa introduziu uma rotação conhecida no crânio, foi possível definir computacionalmente uma transformação rígida para dar a mesma orientação às nuvens de pontos adquiridas (secção 6.2.1). Ao longo deste trabalho, a posição e a orientação entre o projector e a câmara não foram fixas, e sempre que se alteravam, recalibrava-se extrinsecamente o *scanner* 3D. Este factor consituiu uma limitação na avaliação de características do *scanner* 3D, como a sua resolução e o seu alcance, que dependem da posição e da orientação dos elementos que o constituem.

O código fonte de base para aquisição de pontos e para calibração do *scanner* 3D - *cvStrucutredLight*-, mostrou-se eficaz e útil no desenvolvimento do algoritmo *myStrucutredLight*, que permitiu a aquisição de pontos tirando partido da plataforma ro-

tativa (secção 4.3). Os algoritmos desenvolvidos para pré-processamento das nuvens de pontos adquiridas pelo *scanner* 3D, constituem métodos eficazes para a remoção de *outliers* e para a sub-amostragem das nuvens. Graças ao pré-processamento das nuvens, o processo de reconstrução de superfícies é facilitado pela diminuição do tempo de processamento das nuvens de pontos, e pelo facto de as nuvens apresentarem menos ruído.

O algoritmo AI (secção 6.2.2), permitiu alinhar e integrar as diferentes nuvens de pontos, originado uma nuvem de pontos global e uma imagem 3D representativas de toda a superfície do objecto analisado. A partir da imagem 3D procedeu-se à reconstrução da superfície do crânio, recorrendo ao algoritmo *Marching Cubes* (secção 6.3). As superfícies obtidas foram então comparadas com uma superfície do crânio obtida por um *scanner* de TC, a fim de se avaliar a sua qualidade (secção 6.4). Verificou-se que todas as superfícies obtidas no final do trabalho apresentam um desfasamento relativamente à superfície de validação. É importante referir que a estrutura do crânio é complexa, com bastantes zonas difíceis de adquirir, constituindo um desafio para qualquer sistema. Um outro factor de grande influência no resultado final é o grande número de parâmetros de entrada dos diferentes algoritmos, que conduz a um problema de optimização para determinar a melhor combinação de parâmetros.

Na tentativa de aumentar a qualidade da superfície obtida foram exploradas algumas alternativas aos algoritmos utilizados, não tendo sido produzida uma melhoria significativa (secção 6.5). Acredita-se que a implementação futura de alterações constribuirá para a obtenção de um resultado final melhor. A definição e a fixação do posicionamento e da orientação entre o projector e a câmara tornarão possível determinar e avaliar as características do *scanner* 3D, como a sua resolução e o seu alcance. Através do reforço da estrutura da plataforma rotativa será possível evitar oscilações, ainda que mínimas, do objecto em estudo. A conjugação destas duas alterações permitirá definir mais precisamente a transformação rígida para dar a mesma orientação às nuvens de pontos adquiridas. Este factor poderá contribuir para um melhor alinhamento das nuvens e poderá aumentar a qualidade da superfície final.

Numa tentativa de evoluir o trabalho já realizado poder-se-ia criar uma estrutura móvel contendo o projector de vídeo, a câmara, e um terminal móvel com giroscópio. A estrutura seria semelhante às estruturas na secção 2.6, e permitiria analisar objectos que não pudessem ser colocados na plataforma rotativa. Através do giroscópio obter-se-iam as transformações que traduzissem o movimento da estrutura em torno do objecto em análise. Tal como foi feito neste trabalho, essas transformações seriam

aplicadas automaticamente para dar a mesma orientação às nuvens de pontos.

Bibliografia

[1] L. A. Albuquerque and J. M. S. T. Motta, "Implementation of 3D shape reconstruction from range images for object digital modeling," *ABCM Symposium Series in Mechatronics*, vol. 2, pp. 81–88, 2006.

[2] M. J. Thali, S. Ross, L. Oesterhelweg, S. Grabherr, U. B. S. Naether, C. Jackowski, S. Bolliger, P. Vock, and A. C. R. Dirnhofer, *Virtopsy, Working on the Future of Forensic Medicine*, vol. 17, pp. 7–12. Rechtsmedizin, 2007.

[3] M. J. Thali, M. Brauna, and R. Dirnhofer, "Optical 3D surface digitizing in forensic medicine: 3D documentation of skin and bone injuries," *Forensic Science International*, vol. 137, pp. 203–208, July 2003.

[4] R. Siderits, J. Birkenstamm, F. Khani, EvitaSadamin, and J. Godyn, "Three-Dimensional Laser Scanning of "Crime Scene Gum"as a Forensic Method Demonstrating the Creation of Virtual Tooth Surface Contour and Web-Based Rapid Model Fabrication," *Forensic Science Communications*, vol. 12, Number 2, Abril 2010.

[5] T. Lerch, M. MacGillivray, and T. Domina, "3D Laser Scanning: A model of Multidisciplinary Research," *Journal of textile and apparel, technology and management*, vol. 5, Issue 4, 2007.

[6] L. Nishida, "Estudo de técnicas de processamento de imagens aplicadas a um active range finder com projeção de luz estruturada na forma de listras paralelas verticais." Laboratório de Técnicas inteligentes, Universidade de São Paulo.

[7] F. Blais, "Review of 20 years of range sensor development," *Journal of Electronic Imaging*, vol. 13, pp. 231–240, January 2004.

[8] C. Rocchini, P. Cignoni, C. Montani, P. Pingi, and R. Scopigno, "A low cost 3D scanner based on structured light," *Computer Graphics Forum*, vol. 20, pp. 299–308, September 2001.

[9] B. L. Curless, *New methods for surface reconstruction from range images*. PhD thesis, Stanford University, June 1997.

[10] D. Lanman and G. Taubin, "Build your own 3D scanner: 3D photography for beginners," in *SIGGRAPH '09: ACM SIGGRAPH 2009 courses*, pp. 1–87, ACM, 2009.

[11] L. Truppia, "Scansione di oggetti 3D: guida e metodologica e casi di studio," Master's thesis, Università degli studi di catania - Facoltà di scineze Matematiche, Fisichee Naturali, 2006/2007.

[12] D. M. Orozco, "3D-Object Modeling For Computer Games." Presentation at Computer Seminar, Ulm University, 2010.

[13] A. Marbs, " Experiences with Laser Scanning AT i3mainz." Institute for Spatial Information and Surveying Technology, FH Mainz, University of Applied Sciences, Holzstrasse 36, 55116 Mainz, Germany.

[14] G. T. Herman, *Fundamentals of Coomputerized Tomography: Image Reconstruction from Projections*. Springer, 2nd ed., November 2009.

[15] D. J. Brenner and E. J. Hall, "Computed Tomography - An Increasing Source of Radiation Exposure," *The New England Journal of Medicine*, vol. 357, pp. 2277–2284, November 2007.

[16] K. Kolev, T. Brox, and D. Cremers, "Robust Variational Segmentation of 3D Objects from Multiple Views," *DAGM 2006*, vol. 4174, pp. 688-697, pp. 688–697, 2006.

[17] A. C. R. Paiva, "Aquisição de Informação Tridimensional com Luz Estruturada Codificada," Master's thesis, Faculdade de Engenharia da Universidade do Porto, Novembro 1997.

[18] D. Scharstein, *View synthesis using stereo vision*. PhD thesis, Factulty of Graduate School of Cornell University, January 1997.

[19] P. Cignoni and R. Scopigno, "Sampled 3D Models for CH Applications: A viable and Enabling New Medium or Just a Tecnhological Exercise?," *Journal on Computing and Cultural Heritage*, vol. 1, Issue 1, June 2008.

[20] T. C. S. Azevedo, J. M. R. S. Tavares, and M. A. P. Vaz, "External Anatomical Shapes Reconstruction from Turntable Image Sequences using a Single of-the-shelf Camera," *Electronic Letters on Computer Vision and Image Analysis*, vol. 7, no. 2, pp. 22–34, 2008.

[21] J. A. Strickon, "Design and HCI Applications of a Low-Cost Scanning Laser Rangefinder," Master's thesis, Massachusetts Institute of Technology, June 1999.

[22] A. Georgopoulos, C. Ioannidis, and A. Valanis, "Assessing the performance of a structured light scanner," *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XXXVIII, Part 5, 2010.

[23] F. Bernardini and H. Rushmeier, "The 3D Model Acquisition Pipeline," vol. 21, pp. 149–172, 2002.

[24] D. Akca, A. Gruen, Z. Alkis, N. Demir, B. Breuckmann, I.Erduyan, and E. Nadir, "3D Modeling of the Weary Herakles Statue with a Coded Stuctured Light System," *ISPRS Commission V Symposium 'Image Engineering and Vision Metrology'*, vol. XXXVI, pp. 14–19, Septmeber 2006.

[25] D. An, A. Woodward, P. Delmas, and C.-Y. Chen, "Comparison of Active Structure Lighting Mono and Stereo Camera Systems: Application to 3D Face Acquisition," in *Image and Vision Computing New Zealand Conference (IVCNZ)*, (New Zealand), pp. 195–200, 2005.

[26] J. Pages and J. Salvi, "Coded light projection techniques for 3D reconstruction," *J3eA, Journal sur l'enseignement des sciences et technologies de l'informationet des systèmes*, vol. 1, Hors-Série 3, 2005.

[27] J. Salvi, J. Pagès, and J. Batlle, "Pattern codification strategies in structured light systems," *Pattern Recognition*, vol. 37, pp. 827–849, 2004.

[28] J. Posdamer and M. Altschuler, "Surface measurement by space encoded projected beam systems," *Computer Graphics and Image Processing*, vol. 18, no. 47, pp. 1–17, 1982.

[29] S. Inokuchi, K. Sato, and F. Matsuda, "Range imaging system for 3-D object recognition," in *International Conference on Pattern Recognition*, pp. 806–808, 1984.

[30] W. A. Kalender, "X-Ray Computed Tomography," *PHYSICS IN MEDICINE AND BIOLOGY*, vol. 51, pp. R29–R43, June 2006.

[31] P. Vanezis, M. Vanezis, G. McCombe, and T. Niblet, "Facial reconstruction using 3-D computer graphics," *Forensic Science International*, vol. 108, pp. 81–95, 2000.

[32] D. L. Esme, A. Bucksch, and W. H. Beekman, "Three-Dimensional Laser Imaging as a Valuable Tool for Specifying Changesin Breast Shape After Augmentation Mammaplasty," *International Society of Aesthetic Plastic Surgery*, vol. 33, pp. 191–195, November 2009.

[33] M. Y. Hajeer, D. T. Millet, A. F. Ayoub, and J. P. Siebert, "Applications of 3D imaging in orthodontics: Part I and Part II," *Journal of Orthodontics*, vol. 31, pp. 62–70, March 2004.

[34] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, and et. al., "The Digital Michelangelo Project: 3D Scanning of Large Statues," *Proc. SIGGRAPH*, July 2000.

[35] A. Yao, "Applications of 3D scanning and reverse engineering techniques for qualitycontrol of quick response products," *The International Journal of Advanced Manufacturing Technology*, vol. 26, pp. 1284–1288, June 2004.

[36] N. Apuzzo, "3D Body Scanning Technology for Fashion and Apparel Industry," *SPIE*, vol. 6491, 2007.

[37] Polhemus, "FastSCAN^{TM}." http://www.polhemus.com/?page=Scanning_Fastscan.

[38] GOM, "ATOS Compact." http://www.gom.com/metrology-systems/system-overview/atos-compact-scan.html.

[39] FARO, "Focus 3D." http://www.faro.com/focus/uk.

[40] Superfície, "Scan 3D." http://www.superficie.pt/index.php?option=com_content&view=article&id=93&Itemid=64&lang=pt.

[41] S. Tornicasa and E. Vezzetti, "Feasibility study of a reverse engineering system benchmarking." Proceedings del Congreso Internacional Conjunto XVII INGEGRAPH XV ADM, June 2005.

[42] Z. Zhang, "A Flexible New Technique for Camera Calibration," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[43] R. Carey, G. Bell, and C. Marrin, "ISO/IEC 14772-1:1997 Virtual Reality Modeling Language (VRML97)," 1997.

[44] G. Taubin, W. Horn, F. Lazarus, and J. Rossignac, "Geometry Coding and VRML," in *Proceedings of the IEEE*, pp. 1228–1243, IEEE, 1998.

[45] Kitware, *VTK User's Guide*. Kitware, Inc., 11 ed., Março 2010.

[46] R. Kolluri, J. R. Shewchuk, and J. F. O'Brien, "Spectral surface reconstruction from noisy point clouds," in *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, SGP '04, (New York, NY, USA), pp. 11–21, ACM, 2004.

[47] F. Remondino, "From Point Cloud to Surface: The Modeling and Visualization Problem," *Information Sciences*, vol. XXXIV-5/W10, February 2003.

[48] P. J. Besl and N. D. McKay, "A method for registration of 3D shapes," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 14, pp. 239–256, February 1992.

[49] Z. Zhang, "Iterative Point Matching for Registration of Free-From Curves and Surfaces," *International Journal of Computer Vision*, vol. 13, no. 2, pp. 119–152, 1994.

[50] S. Rusinkiewicz and M. Levoy, "Efficient Variants of the ICP Algorithm," in *International Conference on 3D Digital Imaging and Modeling*, pp. 145–152, 2001.

[51] T. Jost and H. Hügli, "Fast ICP algorithms for the registration of 3D data," in *3D Modeling 2003*, 2003.

[52] L. Ibanez, W. Schroeder, L. Ng, and J. Cates, "The ITK Software Guide ," pp. 36 – 43, 2005.

[53] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," *SIGGRAPH Comput. Graph.*, vol. 21, pp. 163–169, Aug. 1987.

[54] Kitware Inc., "Exemplos de código VTK em C++." http://www.vtk.org/Wiki/VTK/Examples/Cxx.

[55] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*, vol. 1. O'Reilly Media, October 2008.

[56] W. J. Schroeder, K. M. Martin, and W. E. Lorensen, "The Design and Implementation of an Object-Oriented Toolkit for 3D Graphics and Visualization," 1996.

Apêndice A

Material Utilizado

A.1 Desenvolvimento da Plataforma Rotativa

Nesta secção são apresentados os materiais utilizados para desenvolver a plataforma rotativa.

- Arduino Duemilanove
  http://www.arduino.cc/

  O Arduino é uma placa controladora baseada no microcontrolador ATmega328 com ligação USB, para alimentação e envio/recepção de dados. As características técnicas do Arduino Duemilanove encontram-se na tabela A.1.

- Motor de passos PM55L-048
  http://www.datasheetarchive.com/PM55L-048-datasheet.html
  O motor PM55L-048 é um motor de passos unipolar, do tipo PM (tabela A.2). Este motor é capaz de dar passos com uma resolução de 7.5°/passo, dando o total de 48 passos por revolução. O motor seleccionado tem cinco fios, sendo que um é para a alimentação e os restantes são para o seu controlo através do envio de sinais. Este motor tem ímanes permanentes (PM) na sua estrutura e o rotor é magnetizado com pólos norte e sul situados numa linha paralela ao eixo de rotação do motor.

- ULN2003
  http://www.st.com/stonline/products/literature/ds/5279.pdf
  O ULN2003 é um circuito integrado que contém sete pares darlington, sendo capaz de fornecer até 500 mA e 50 V nas suas saídas. Cada um dos pares pode ser acionado por um nível lógico TTL permitindo o controlo de dispositivos que operam a uma corrente superior à que o arduino é capaz de fornecer.

- Breadboard;

- Cabo USB;

- Leds, cabos de cobre e resistências de 1 kΩ;
- Roda dentada para a engrenagem e parafuso com rosca sem-fim para o eixo da plataforma:
- Acrílico para a estrutura da plataforma.

Tab. A.1: Características da placa microcontroladora Arduino Duemilanove.

Características	
Microcontrolador	ATmega328
Tensão de funcionamento	5V
Tensão de entrada (recomendada)	7-12V
Tensão de entrada (limite)	6-20V
Pinos I/O digitais	14 (dos quais 6 provêm saídas PWM)
Pinos de entrada analógicos	6
CC por pino I/0	40 mA
CC para pino de 3.3V	50 mA
Memória flash	32 KB (2 KB usados pelo bootloader)
SRAM	2 KB
EEPROM	1 KB
Freq. relógio	16 MHZ

Tab. A.2: Motor de passos PM55L-048.

Características	
Nº. de passos por revolução	48 (7.5º/passo)
N.º de fios	5
Tipo de ímane	Permanente (PM)
Circuito	unipolar

A.2 *Software* e *Toolkits*

Para o desenvolvimento dos algoritmos apresentados neste trabalho, e para a visualização dos resultados obtidos recorreu-se a um conjunto de *software* e *toolkits* gratuítos, descritos nesta secção.

Sistema Operativo

- Windows 7 Home Premium 64 bits

IDE

- Microsoft Visual Studio – VC++ Express Edition
  http://msdn.microsoft.com/en-us/express/future/bb421472
  Visual Studio C++ Express é um ambiente de desenvolvimento integrado gratuíto, desenvolvido pela Microsoft para programação em C++.

- Arduino alpha 0021
  http://www.arduino.cc/en/Main/Software
  Arduino alpha é um ambiente de desenvolvimento integrado, gratuíto, que permite desenvolver aplicações para controlo do Arduino. É escrito em Java e baseado nas linguagens de programação Processing (http://www.processing.org/), avr-gcc, e outras. Arduino alpha é desenvolvido e mantido por Massimo Banzi, David Cuartielles, Tom Igoe, Gialunca Martino e David Mellis.

Toolkits

- Open Source Computer Vision library (OpenCV)
  http://opencv.willowgarage.com/wiki/
  OpenCV é uma biblioteca open-source escrita em C e C++ para visão por computador. Originalmente desenvolvida pela Intel em 2000, é distribuída sob licença BSD, funcionando em ambiente Linux, Windows e Mac OS X, e disponibilizando mais de quinhentas funções. Actualmente encontram-se em desenvolvimento interfaces para Python, Ruby, Matlab e outras linguagens de programação. OpenCV fornece uma infraestrutura simples que abrange várias áreas na Visão por Computador, disponibilizando funções para os seguintes ramos: inspecção de produtos na indústria, imagem médica, segurança, interface de utilizador, calibração de câmaras, visão estereoscópica e robótica. Adicionalmente, também disponibiliza uma libraria para *Machine Learning* [55].

- Visualization Toolkit (VTK)
  http://www.vtk.org/VTK/project/about.html
  VTK é uma biblioteca livre para desenvolvimento de sistemas de visualização, gráficos 3D e processamento de imagem. Consiste numa biblioteca de classes C++ e várias camadas de interfaces para Tcl/TK, Java e Python. Esta

biblioteca suporta uma série de algoritmos processamento de imagem e visualização. Actualmente, VTK é utilizado em aplicações comerciais, investigação e desenvolvimento, e encontra-se na base de muitas aplicações de visualização avançada, tais como: ParaView, VisIt, VisTrails, Slicer, MayaVi e OsiriX. VTK é mantido e desenvolvido por utilizadores de todo o mundo, existindo uma mailing list para troca de informação e de ideias entre utilizadores e desenvolvedores [56].

- Insight Segmentation and Registration Toolkit (ITK)
  http://www.itk.org/ITK/project/about.html
  ITK é uma biblioteca open-source para processamento, alinhamento e segmentação de imagem. ITK é implementado em C++ e utiliza o CMake para gerir o processo de configuração. Adicionalmente, permite gerar interfaces entre C++ e linguagens de programação interpretadas como Tcl, Java, Python e (usando CableSwig). Desta forma é possível desenvolver-se software utilizando uma variedade de linguagens de programação. Actualmente, ITK é utilizado, mantido e desenvolvido por utilizadores de todo o mundo, existindo uma *mailing list* para troca de informação e de ideias entre utilizadores e desenvolvedores.

Software

- Paraview v.3.10.0
  http://www.paraview.org/paraview/project/about.html
  Paraview é uma aplicação open-source e multi-plataforma para análise de informação e visualização científica. Paraview disponibiliza ferramentas para análise qualitativa e quantitativa de dados, permitindo a exploração de informação interactivamente a 3D ou de forma programática, através de processamento batch. Paraview funciona em ambientes Windows, Mac OS X, Linux, IBM Blue Gene, Cray Xt3 e em outros sistemas UNIX. Utiliza o VTK como motor de processamento e renderização, e a sua interface de utilização foi escrita utilizando Qt®.